CONTENTS

Page

CHAPTER 11 MESSAGE UTILITY

CHAPTER 12 MONITOR UTILITY (MONITOR)

EXAMPLES

FIGURES

TABLES

PREFACE

This reference manual describes utility programs supported by DIGITAL on the VAX/VMS operating system.

INTENDED AUDIENCE

This manual is intended for users who are already familiar with VAX/VMS system concepts. Use of the various utility programs is appropriate for users at different levels of experience and responsibility. The expected user group for each program is defined below in the chapter summaries.

STRUCTURE OF THIS DOCUMENT

This manual is organized into nineteen chapters and four appendixes.

Each chapter of this manual describes one utility program, except for Chapter 16, which includes two related editors (SLP and SUMSLP). Chapter 7 includes two variants of a disk back-up program (DSC). Each chapter contains a list of the messages issued by the utility. The following are the contents and intended audience of each chapter.

Chapter 1 describes the Accounting Utility. This utility provides listings of selected accounting records and a summary report of selected accounting record items. It includes a sorting facility for producing accounting reports. The Accounting Utility is intended for all system users.

Chapter 2 describes the Authorize Utility. This utility creates a new, or modifies an existing, user authorization file. It is intended for all system users.

Chapter 3 describes the Backup Utility, referred to as BACKUP. This program allows you to back up and restore disk volumes, directories, or individual files. You can also use the Backup Utility to copy disk volumes, directories, or individual files. It is intended for use by all users.

Chapter 4 describes the Bad Block Locator Utility, referred to as BAD. This program determines and records the number and location of bad blocks on block-structured volumes. BAD is intended for use by VAX/VMS system managers, operators, and system programmers.

Chapter 5 describes the Command Definition Utility. This utility provides a means for the user to generate commands. It is intended for all system users.

Chapter 6 describes the Disk Quota Utility. This utility allows the system manager to establish disk quotas. The system automatically records disk usage and enforces quotas during file operations.

Chapter 7 describes the Disk Save and Compress Utilities, referred to as DSC1 and DSC2. (The stand-alone version, DSC-2, is no longer supported.) The DSC programs are used to back up and restore disk volumes that have been formatted and initialized as Files-11 volumes. DIGITAL does not intend to support the use of the DSC utilities on VAX/VMS for VAX/VMS Version 4.0 and beyond. Users should convert to the use of the Backup Utility described in Chapter 3.

Chapter 8 describes the File Transfer Utility, referred to as FLX (and generally pronounced "FILEX"). This program transfers files from one volume to another and performs volume format conversions. FLX is intended for use by all system users.

Chapter 9 describes the Install Utility. This utility allows the system manager to install selected executable and shareable images.

Chapter 10 describes the Librarian Utility, referred to as the Librarian. This program allows you to store useful modules in a central, easily accessible location. It is intended for use by all VAX/VMS users.

Chapter 11 describes the Message Utility. This program allows you to construct your own informational, warning, and error messages, or to customize the messages provided by VAX/VMS. The Message Utility is intended for use by VAX/VMS system programmers and application programmers.

Chapter 12 describes the Monitor Utility. This utility allows the system manager to obtain information on operating system performance data such as I/O statistics, page management statistics, and time spent in each of the processor modes.

Chapter 13 describes the Personal Mail Utility, referred to as MAIL. This program allows users to send messages to one another, within the same system or between any VAX-11 computers that are connected by means of Decnet-VAX. Use of MAIL is appropriate for all system users.

Chapter 14 describes the Phone Utility. This utility allows the .user to talk with other users on their system or any other VAX-11 computer that is connected to their system by means of DECnet-VAX. It is intended for all users.

Chapter 15 describes the RMS Share Utility. This utility is a system management tool that enables VAX-11 Record Management Services (VAX-11 RMS) file-sharing capability and displays figures on allowable and actual file-sharing usage.

Chapter 16 describes two related batch-oriented text editors, SLP and SUMSLP. These editors are used to incorporate changes into source files and to indicate these changes with an audit trail. SLP and SUMSLP are intended for use by all system users.

Chapter 17 describes the SYE Utility. This utility allows the system manager to obtain information on the contents of error logging files.

Chapter 18 describes the System Generation Utility. This utility allows the system manager to perform various functions to ensure proper configuration of the system.

Chapter 19 describes the Verify Utility. This utility checks the readability and validity of Files-11 Structure Level 1 and Level 2 disk volumes and reports errors and inconsistencies to the user. It is intended for all system users.

Appendix A lists the Files-11 structured devices supported by VAX/VMS, with the characteristics and device code of each device. It presents information needed by users of several of the utilities in this manual.

Appendix B describes the media formats used by the Backup Utility to save data. This information is relevant only to the Backup Utility.

Appendix C describes the format of accounting file data. It is applicable to the Accounting Utility.

Appendix D describes the Monitor Utility recording file record formats.

ASSOCIATED DOCUMENTS

To use the utilities described in this document, you should be familiar with the following manuals:

- VAX/VMS Primer
- VAX/VMS Command Language User's Guide
- VAX/VMS Summary Description and Glossary

Some of the utilities require familiarity with disk structures and volume concepts described in the following manuals:

- VAX/VMS System Management and Operations Guide
- Introduction to VAX-11 Record Management Services
- VAX-11 Record Management Services Reference Manual

Some of the utilities run in compatibility mode; readers may wish to consult the VAX-11/RSX-11M User's Guide.

Note that there are other utility programs that run on VAX-11 processors - PATCH and the System Dump Analyzer, for example. These programs are described elsewhere in the VAX-11 documentation set. For a complete list of VAX-11 documents, including a brief description of each, see the VAX-11 Information Directory and Index.

CONVENTIONS USED IN THIS DOCUMENT

The following conventions are observed in this manual:

Convention	Meaning
Uppercase words and letters	Uppercase words and letters, used in examples, indicate that you should type the word or letter exactly as shown.
Lowercase words and letters	Lowercase words and letters, used in format examples, indicate that you are to substitute a word or value of your choice.

Convention	Meaning
Quotation marks Apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.
$ SHOW TIME 05-JUN-1980 11:55:22	Command examples show all output lines or prompting characters that the system prints or displays in black letters. All user-entered commands are shown in red letters.
[ ]	Square brackets indicate that the enclosed item is optional.
{ }	Braces are used to enclose lists from which one element is to be chosen.
...	A horizontal ellipsis indicates that the preceding item(s) can be repeated one or more times.
. . .	A vertical ellipsis indicates that not all of the statements in an example or figure are shown.
RET	A symbol with a 1- to 3-character abbreviation indicates that you press a key on the terminal, for example, RET .
CTRL/X or CTRL/x	The phrase CTRL/x indicates that you must press the key labeled CTRL while you simultaneously press another key, for example CTRL/C, CTRL/Y, CTRL/O. In examples, this control key sequence is shown as ^x, for example ^C, ^Y, ^O, because that is how the system echoes control key sequences.

Unless otherwise noted, all numeric values are represented in decimal notation.

Unless otherwise specified, you terminate commands by pressing the RETURN key.

Where the term file-spec is used in this document, it refers to a file specification constructed according to the following definitions, with the format:

```
     node::device:[directory]filename.type;version
```

The punctuation marks (colons, brackets, period, semicolon) are required syntax that separate the various components of the file specification.

node

A network node name. This is applicable only to systems that support DECnet-VAX.

device

The device on which the file is stored or is to be written.

directory

The name of the directory under which the file is cataloged on the device specified. You can delimit the directory name with square brackets, as shown, or with angle brackets (<>).

filename

The file by a name of up to 9 alphanumeric characters.

type

The type of data in the file; type can be up to 3 alphanumeric characters.

version

The version of the file. Versions are identified by a decimal number, which is increased by 1 each time a new version of the file is created. Either a semicolon or a period can be used to separate type and version.

You need not always state all elements of a file specification explicitly. Frequently, only the file name is required. If you omit other parts of the file specification, a default value is provided as described in the following table.

Optional Element	**Default Value**
node	Local network node
device	User's current default device
directory	User's current default directory
type	Depends on use, as described with the various utilities
version	Input: highest existing version Output: highest existing version plus 1

Any variations in file specifications that a utility program requires are stated in the description of that utility.

SUMMARY OF TECHNICAL CHANGES

Modifications to this manual reflect the changes described below to VAX/VMS for Version 3.0.

1.0 SUMMARY OF CHANGES

Reference information for the following utilities has been moved from the former VAX/VMS System Manager's Guide to this manual:

- Authorize Utility (see Chapter 2)
- Disk Quota Utility (see Chapter 6)
- Install Utility (see Chapter 9)
- RMS Share Utility (see Chapter 15)
- SYE Utility (see Chapter 17)
- System Generation Utility (see Chapter 18)

In moving the descriptions of the utilities, all error messages were moved to the VAX/VMS System Messages and Recovery Procedures Manual.

In addition, the Display Utility is replaced functionally in Version 3.0 by the new Monitor Utility. See the description below of the Monitor Utility for details of the enhancements offered by Monitor. Chapter 12 describes the Monitor Utility, while Appendix D presents the formats of the Monitor Recording File Records.

Also new with Version 3.0 is the Accounting Utility (ACCOUNTING), which provides reports from the data in the accounting log file, ACCOUNTING.DAT. See Chapter 1 for details about the Accounting Utility, and see Appendix C for a summary of all record formats in the revised accounting log file.

1.1 Accounting Utility (ACCOUNTING)

The following summary of features briefly introduces the new Accounting Utility:

- Provides brief listings of selected accounting records
- Provides full listings of selected accounting records
- Produces a binary copy of selected accounting records
- Produces a summary report of selected items from selected accounting records
- Includes the ability to sort accounting records

1.2 Authorize Utility

The Authorize Utility has undergone a number of modifications. AUTHORIZE permits you to designate primary and secondary days of the week for each user for login purposes. You can then restrict the hours that the user can log in on both primary and secondary days. Furthermore, you can restrict whether or not the user can log in over the network or by a dial-up line on primary and secondary days. These changes have been implemented through five new qualifiers (see below).

The new commands are:

- COPY -- Creates a system UAF record that duplicates an existing record.
- RENAME -- Changes the name of a user in a current system UAF record.

The new qualifiers are:

- /ENQLM -- Limit for the number of locks that can be queued at one time.
- /MAXACCTJOBS -- (Formerly /MAXGRPJOBS, this qualifier is not currently implemented.)
- /MAXJOBS -- Maximum number of top-level processes, not including subprocesses, that can be active at one time.
- /PRIMEDAYS -- Defines the primary and secondary days of the week for login purposes.
- /PFLAGS -- Controls whether logins via dial-up lines or over the network are allowed on primary days.
- /P_RESTRICT -- Controls which hours of each primary day are allowed for logins.
- /SFLAGS -- Controls whether logins via dial-up lines or over the network are allowed on secondary days.
- /S_RESTRICT -- Controls which hours of each secondary day are allowed for logins.
- /WSEXTENT -- Default working set extent.

The new keywords for the /FLAGS qualifier are:

- CAPTIVE -- Disables CTRL/Y interrupts, prohibits the use of the DCL command SET PASSWORD, and prohibits user specification of a default CLI.
- DISNEWMAIL -- Suppresses the announcements of new mail at login time.
- DISWELCOME -- Suppresses the login message "Welcome to ..."
- DISUSER -- Prevents the user from logging in.

The new AUTHORIZE features include:

- The asterisk (*) and percent sign (%) wild card characters are permitted with the SHOW, LIST, and MODIFY commands.
- The HELP command now employs the new system help facility, with interactive features.

- The display resulting from the SHOW command now includes fields for the new qualifiers and keywords shown above.

1.3 Backup Utility (BACKUP)

Several changes have been made and new qualifiers added to this chapter, which is now Chapter 3.

Full disk volumes, including system disks, can now be saved or copied using the /IMAGE qualifier. With this new feature, BACKUP is now more effective than the DSC utilities in saving and copying full disk volumes.

Sequential disk save sets allow you to create a save set on a series of disk volumes, much like creating a save set on a series of tape volumes.

Save sets can be created on, and restored from, a remote VAX/VMS node.

The /LIST qualifier can be used in conjunction with any BACKUP operation (save, restore, copy, or compare).

Stand-alone backup allows users to perform image BACKUP operations offline.

The new qualifiers are:

- /DELETE -- Delete files when they have been successfully saved or copied.
- /IGNORE=INTERLOCK -- Save or copy files that are open for writing.
- /IGNORE=NOBACKUP -- Save or copy the contents of files that are marked with the NOBACKUP flag.
- /IMAGE -- Perform an image operation. An image operation processes an entire volume or volume set and allows users to create a funtionally equivalent copy of the input volume.
- /INCREMENTAL -- Restore an image save set and a series of incremental save sets such that the output volume will contain the same files that it contained when the most recent incremental back-up was performed.
- /INITIALIZE -- Initialize an output disk volume when restoring or copying with the /IMAGE qualifier. Initialization data is obtained from the input volume. The /NOINITIALIZE qualifier directs BACKUP to use initialization data from the output volume.
- /JOURNAL -- Record the names of files saved by BACKUP; journal file can be searched for the save set containing a specific file.
- /VOLUME -- Process a specific disk in a disk volume set (only valid with the /IMAGE qualifier).
- /OWNER_UIC -- Assign owner UIC to an output save set.
- /PROTECTION -- Assign protection to an output save set.

1.4 Command Definition Utility

This new chapter (Chapter 5) describes the Command Definition Utility, which allows you to add new commands to the DCL Command Language.

1.5 Disk Save And Compress Utilities

This chapter contains changes in the use and support of the Disk Save and Compress Utilities. The stand-alone version, DSC-2, is no longer supported. Furthermore, DIGITAL does not intend to support the other two versions (DSC1 and DSC2) on VAX/VMS for VAX/VMS Version 4.0 and beyond. Users should convert to the use of the `Backup Utility described in Chapter 3.

1.6 Install Utility

The following changes have been made to INSTALL:

- Support for rooted directories has been added.
- The /EXIT qualifier can be used to exit.
- The /HELP qualifier can be used to obtain interactive help.
- Error reporting using the message facility has been enhanced. The names of files that are not found are reported. The offending privilege is reported if there is an error specifying privilege names. (The privilege names are displayed by /HELP.)
- The /LIST/FULL qualifiers decode the privilege mask.

1.7 Librarian Utility

Several new qualifiers and routines have been added to this chapter, which is now Chapter 10. The new qualifiers are:

- /BEFORE -- Specifies that only those modules dated earlier than a particular time be listed.
- /COMPRESS -- Requests the LIBRARY command to either recover unused space in the library resulting from module deletion, or reformat a library created by the VAX/VMS Version 1.0 or Version 2.0 Librarian into Version 2.0 or Version 3.0 format.
- /CREATE -- Requests the LIBRARY command to create a new library.
- /FULL -- Requests a full description of each module in the module name table.
- /HISTORY -- Specifies the record header and update history header format.
- /SINCE -- Specifies that only those modules dated later than a particular time be printed.

The new library routines are:

- LBR$FLUSH -- Writes the contents of modified blocks to the library file and returns the virtual memory that contained those blocks.
- LBR$GET_HEADER -- Retrieves information from the library header.
- LBR$GET_HISTORY -- Retrieves library update history records and calls a user-supplied routine with each record returned.
- LBR$OPEN -- Opens an existing library or creates a new one.
- LBR$OUTPUT_HELP -- Retrieves help text from an explicitly named library or from user-supplied default libraries and optionally prompts the user for additional help queries.
- LBR$PUT_HISTORY -- Inserts a library update history record.
- LBR$RET_RMSSTV -- Returns the last VAX-11 RMS status value.
- LBR$SET_LOCATE -- Sets Librarian subroutine record access to locate mode.
- LBR$SET_MOVE -- Sets Librarian subroutine record access to move mode.

This chapter also contains the new LIBRARY command help module.

1.8 Monitor Utility

The new Monitor Utility (MONITOR) is a functional replacement for the Display Utility (DISPLAY).

For Version 2.0 DISPLAY, performance data was grouped into classes, collected at a user-specified interval and formatted for screen presentation to a VT52, VT55, or VT100 terminal.

Version 3.0 MONITOR provides significant improvements over Version 2.0 DISPLAY in the flexibility of processing and presenting VAX/VMS performance data. These improvements include:

- Presentation of data to the user's terminal, independent of terminal type, including hardcopy and printable file.
- Recording of any or all classes of performance data in a disk file and subsequent "playback" of those classes to the user's terminal.
- Data reduction of a recorded file by providing a rerecording capability for any subset of recorded classes, over any time segment within the recorded segment, at an interval independent of the recorded interval.
- Ability to obtain certain statistical information (such as average value, maximum value, and minimum value) for any range of recorded data.
- Concurrent recording and display outputs.

You will find the following equivalences:

DISPLAY Command	MONITOR Class
(None)	DECNET
FCP	FCP
IORATES	IO
(None)	LOCK
M2	MODES
M5	(None)
PAGE	PAGE
POOL	POOL
S2	STATES
S5	(None)
TOPUSERS	PROCESSES/TOPCPU
USERS	PROCESSES

1.9 Personal Mail Utility (MAIL)

This chapter, which is now Chapter 13, has been completely rewritten and reorganized. Two new commands have been added:

- QUIT -- Cancels message deletions and exits from MAIL.
- SEARCH -- Searches for a message that contains specified text.

1.10 Phone Utility

This new chapter (Chapter 14) describes the Phone Utility, which allows you to talk to other users on your system or any other VAX-11 computer that is connected to your system by means of DECnet-VAX. It was designed to closely simulate the features of a real telephone, including the hold button, conference calls, and telephone directories.

1.11 SYE Utility

The following new category type qualifiers are available with the SYE Utility:

- /BU -- Bugcheck entries.
- /DA -- Device attention entries.
- /DE -- Device error bit(s) set entries.
- /DT -- Device I/O timeouts.
- /ME -- Memory errors detected by the scanning code and interrupts and fatal memory errors logged by the mcheck code.
- /UN -- Unknown device entries.

New features include:

- Support for new devices.
- MAILBOX is a valid input file name (for users with the DIAGNOSE user privilege) that causes SYE to report error log entries as they occur on the system.
- Asterisk (*) wild card character allowed in device specifications.
- More than one kind of device can be specifically requested, or excluded, when specifying the qualifiers /BU, /DA, /DE, and /DT.

There has been a change in the procedure to restart ERRFMT. You simply enter:

```
$ @SYS$SYSTEM:STARTUP ERRFMT
```

1.12 System Generation Utility

The new SYSGEN command is:

- CONFIGURE -- Requests UNIBUS device names and outputs the set of CSR and vector addresses that AUTOCONFIGURE uses.

New SYSGEN command qualifiers are as follows:

- AUTOCONFIGURE/EXCLUDE -- Identifies one or more devices not to be autoconfigured.
- AUTOCONFIGURE/LOG -- Lists the controller and its units on the SYS$OUTPUT device after successful autoconfiguration.
- SET/OUTPUT -- Defines an output file for the session.
- SHOW/ADAPTER -- Lists all the nexus numbers and generic names.
- SHOW/CONFIGURATION -- Shows devices by name, number of units, nexus number, adapter type as well as CSR and vector addresses.
- SHOW/DRIVER -- Displays information on connected devices and loaded drivers.
- SHOW/HEX -- Displays the parameter values in hexadecimal.
- SHOW/SCS -- Displays the System Communication Services parameters.
- SHOW/TTY -- Displays the terminal parameters.
- SHOW/UNIBUS -- Displays the addresses in UNIBUS I/O space that can be addressed.

New features include:

- Help provided by the HELP command is interactive.
- PARAMETERS is a valid keyword for the HELP command; it displays information on the system parameters.

- WRITE ACTIVE and WRITE CURRENT send a message to OPCOM and log the event.

1.13 Verify Utility

This new chapter (Chapter 19) describes the Verify Utility (ANALYZE/DISK_STRUCTURE), which replaces the File Structure Verification Utilities (VFY1 and VFY2). The Verify Utility checks the readability and validity of Files-11 Structure Level 1 and Files-11 Structure Level 2 disk volumes and reports errors and inconsistencies to the user.

CHAPTER 1

ACCOUNTING UTILITY (ACCOUNTING)

The Accounting Utility (ACCOUNTING) uses the data in one or more previously recorded copies of the system's accounting log file to produce new files or reports. You can use the accounting reports as system management tools to learn more about the ways the system is used, how it performs, and in some cases, how particular users use the system. The reports can also provide a means of billing users for system resources.

This chapter describes ACCOUNTING's features and applications. For additional information on how to interpret the information the Accounting Utility provides, see the VAX/VMS System Management and Operations Guide. You may also want to examine Appendix C, Format of Accounting Log File Data, to learn more about the exact data in the accounting log file.

The Accounting Utility enables you to obtain information on operating system usage. It processes data by selection and/or sorting to produce four forms of optional output:

- A brief listing of selected records
- A full listing of selected records
- A binary copy of selected records
- A summary report of selected items from selected records

These forms of output can be directed to a terminal for display or to a disk or tape file.

ACCOUNTING processes accounting log data by sharing the currently open log file on a running system or by receiving as input a previously recorded accounting file. All input data must be in binary format.

Use of ACCOUNTING requires read access to the input accounting file.

1.1 INVOKING AND TERMINATING ACCOUNTING

The following DCL command invokes the utility:

```
ACCOUNTING  [file-spec[,...]]
```

Each time you issue the DCL command ACCOUNTING, the utility executes a single ACCOUNTING request. Generally, each ACCOUNTING request runs until it completes. However, you can press CTRL/Y to terminate ACCOUNTING earlier. Pressing CTRL/Y terminates ACCOUNTING under normal conditions, ensuring that all files are handled properly.

1.2 COMMAND SUMMARY

This section presents a summary of the ACCOUNTING command format.

1.2.1 Command Format

The ACCOUNTING command adheres to the syntax and grammar rules of all DCL commands. These rules appear in the VAX/VMS Command Language User's Guide. The ACCOUNTING command has the following format:

```
     ACCOUNTING  [file-spec[,...]]
```

Command Qualifiers	Defaults
/[NO]ACCOUNT=(["-",]account-name[,...])	/NOACCOUNT
/[NO]ADDRESS=(["-",]node-address[,...])	/NOADDRESS
/BEFORE[=time]	current time
/[NO]BINARY	/NOBINARY
/[NO]ENTRY=(["-",]queue-entry[,...])	/NOENTRY
/[NO]FULL	/NOFULL
/[NO]IDENTIFICATION=(["-",]process-id[,...])	/NOIDENTIFICATION
/[NO]IMAGE=(["-",]image-name[,...])	/NOIMAGE
/[NO]JOB=(["-",]job-name[,...])	/NOJOB
/[NO]LOG	/NOLOG
/[NO]NODE=(["-",]node-name[,...])	/NONODE
/[NO]OUTPUT[=file-spec]	/OUTPUT=SYS$OUTPUT
/[NO]OWNER=(["-",]owner-process-id[,...])	/NOOWNER
/[NO]PRIORITY=(["-",]priority[,...])	/NOPRIORITY
/[NO]PROCESS=(["-",]process-type[,...])	/NOPROCESS
/[NO]QUEUE=(["-",]queue-name[,...])	/NOQUEUE
/[NO]REJECTED[=file-spec]	/NOREJECTED
/[NO]REMOTE_ID=(["-",]remote-id[,...])	/NOREMOTE_ID
/[NO]REPORT[=(report-item[,...])]	/NOREPORT
/[NO]SINCE[=time]	/NOSINCE
/[NO]SORT[=([-]sort-item[,...])]	/NOSORT
/[NO]STATUS=(["-",]exit-status[,...])	/NOSTATUS
/[NO]SUMMARY=(summary-item[,...])	/NOSUMMARY
/[NO]TERMINAL=(["-",]terminal-name[,...])	/NOTERMINAL
/[NO]TITLE=title	/NOTITLE
/[NO]TYPE=(["-",]record-type[,...])	/NOTYPE
/[NO]UIC=(["-",]uic[,...])	/NOUIC
/[NO]USER=(["-",]user-name[,...])	/NOUSER

1.2.2 Command Parameters

ACCOUNTING does not prompt. You can optionally specify one or more files as input files, as follows:

file-spec[,...]

> If one or more input files are specified, data is processed from a previously created file; otherwise, data is processed from the current accounting log, SYS$MANAGER:ACCOUNTNG.DAT. If you specify more than one file name, separate them with commas.
>
> Wild card characters are allowed in the file specifications.

1.2.3 Command Qualifiers

The ACCOUNTING command processes one or more files of VAX/VMS accounting log data. It processes the requested data for the interval specified by the /SINCE and /BEFORE qualifiers (either explicitly or implicitly), unless interrupted. It is capable of producing many combinations of output. It formats and directs the output based on the specification of the /OUTPUT, /BINARY, /FULL, /REJECTED, and /SUMMARY qualifiers.

By default, the output is directed to the SYS$OUTPUT device. However, you can specify an output file with the /OUTPUT qualifier. You can further specify whether the output should be in binary or ASCII format with the /BINARY qualifier. If you specify /BINARY, a binary accounting file is produced. However, if you specify /FULL or /SUMMARY, or assume the default, an ASCII file is produced, containing summary statistics for all requested data over the duration of the ACCOUNTING request.

You can select records based on fields in the records and their values. A large number of qualifiers define the field names for selection. Every time you select records, you create a group of one or more unselected records that you can optionally store in a binary output file with the /REJECTED qualifier. You can also sort the records prior to listing them.

Section 1.4 describes the modes of operation and presents examples of ACCOUNTING commands.

/ACCOUNT=["-",]account-name[,...]
/NOACCOUNT

Controls whether only those records matching the specified account-name are selected. The account-name matches the account name as specified in the user authorization file.

When you specify the /ACCOUNT qualifier, you must specify at least one account-name. If you specify more than one account-name, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any account-name in the list.

If you omit the qualifier or specify /NOACCOUNT, the account-name is not used to select records.

/ADDRESS=["-",]node-address[,...]
/NOADDRESS

Controls whether only those records matching the specified node-address are selected. The node-address is a unique numeric identifier for DECnet nodes.

When you specify the /ADDRESS qualifier, you must specify at least one node-address. If you specify more than one node-address, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any node-address in the list.

If you omit the qualifier or specify /NOADDRESS, the node-address is not used to select records.

/BEFORE[=time]

Controls whether only those records dated earlier than the specified time are selected. You can specify an absolute time, delta time, or a combination of the two. Observe the syntax rules for date and time described in the VAX/VMS Command Language User's Guide.

If you specify /BEFORE without the time or omit the qualifier, the current date and time is used by default.

/BINARY
/NOBINARY

Controls whether output is formatted in either binary or ASCII.

When /BINARY is specified, the output file, specified using the /OUTPUT qualifier, contains image copies of the input records. If you specify /NOBINARY or omit the qualifier, the output file contains formatted ASCII records.

/ENTRY=(["-",]queue-entry[,...])
/NOENTRY

Controls whether only those records matching the specified queue-entry are selected. The queue-entry is a unique numeric identifier assigned to entries in device and batch queues.

When you specify the /ENTRY qualifier, you must specify at least one queue-entry. If you specify more than one queue-entry, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any queue-entry in the list.

If you specify /NOENTRY or omit the qualifier, the queue-entry is not used to select records.

/FULL
/NOFULL

Controls whether a full format is used in ASCII displays. By default, records are displayed in the brief format. You must specify /FULL to have the full contents of each selected record displayed.

The /FULL, /SUMMARY, and /BINARY qualifiers are conflicting qualifiers.

If you specify /NOFULL or omit the qualifier, records are displayed in the brief format. For a description of the display formats, see Section 1.3.

/IDENTIFICATION=(["-",]process-id[,...])
/NOIDENTIFICATION

Controls whether only those records matching the specified process-id are selected.

When you specify /IDENTIFICATION, you must specify at least one process-id. If you specify more than one process-id, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any process-id in the list.

If you specify /NOIDENTIFICATION or omit the qualifier, the process-id is not used to select records.

/IMAGE=(["-",]image-name[,...])
/NOIMAGE

Controls whether only those records matching the specified image-name are selected. Specify only the file name portion of the image file specification, such as EDT.

When you specify /IMAGE, you must specify at least one image-name. If you specify more than one image-name, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any image-name in the list.

If you specify /NOIMAGE or omit the qualifier, the image-name is not used to select records.

/JOB=(["-",]job-name[,...])
/NOJOB

Controls whether only those records matching the specified job-name are selected. A job-name is assigned to an entry in a device or batch queue.

When you specify /JOB, you must specify at least one job-name. If you specify more than one job-name, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any job-name in the list.

If you specify /NOJOB or omit the qualifier, the job-name is not used to select records.

/LOG
/NOLOG

Controls whether informational messages (input file names, selected record counts, rejected record counts) are displayed on the current SYS$OUTPUT device. By default, these messages are not displayed. If more than one input file is specified in an ACCOUNTING command with the /LOG qualifier, there is one logging message for each file and a total is provided.

/NODE=(["-",]node-name[,...])
/NONODE

Controls whether only those records matching the specified DECnet node-name are selected. Colons (:) are not allowed in the node-name specification, so you would select the records for a node such as AURA with /NODE=AURA.

When you specify /NODE, you must specify at least one node-name. If you specify more than one node-name, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any node-name in the list.

If you specify /NONODE or omit the qualifier, the node-name is not used to select records.

/OUTPUT[=file-spec]
/NOOUTPUT

Controls whether selected records are output to a specified file.

The /OUTPUT qualifier allows you to specify the name of the file that is to contain the selected records. If you omit the device or directory specification, the current device and default directory are used. If you omit the file name, then the file name of the input file is used. If you omit the file type and the output is ASCII (/NOBINARY), the default file type is LIS. If you omit the file type and the output is binary (/BINARY), the default file type is DAT.

If you omit the qualifier, selected records are output to the current SYS$OUTPUT device.

/OWNER=(["-",]owner-process-id[,...])
/NOOWNER

Controls whether only those records matching the specified owner-process-id are selected. Owner-process-ids are only present in subprocesses to specify the process-id of their owner process.

When you specify /OWNER, you must specify at least one owner-process-id. If you specify more than one owner-process-id, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any owner-process-id in the list.

If you specify /NOOWNER or omit the qualifier, the owner-process-id is not used to select records.

/PRIORITY=(["-",]priority[,...])
/NOPRIORITY

Controls whether only those records matching the specified base process priority are selected.

When you specify /PRIORITY, you must specify at least one priority. If you specify more than one priority, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any priority in the list.

If you specify /NOPRIORITY or omit the qualifier, the priority is not used to select records.

/PROCESS=(["-",]process-type[,...])
/NOPROCESS

Controls whether only those records matching the specified process-type are selected.

When you specify /PROCESS, you must specify at least one process-type. If you specify more than one process-type, separate them with commas and enclose the list in parentheses.

You can specify any of the following process types:

Keyword	Meaning
BATCH	Batch process
DETACHED	Detached process
INTERACTIVE	Interactive process
NETWORK	Network processs
SUBPROCESS	Subprocess

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any process-type in the list.

If you specify /NOPROCESS or omit the qualifier, the process-type is not used to select records.

/QUEUE=(["-",]queue-name[,...])
/NOQUEUE

Controls whether only those records matching the specified queue-name are selected. A queue-name is a unique identifier for a device or batch queue.

When you specify /QUEUE, you must specify at least one queue-name. If you specify more than one queue-name, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any queue-name in the list.

If you specify /NOQUEUE or omit the qualifier, the queue-name is not used to select records.

/REJECTED[=file-spec]
/NOREJECTED

Controls whether unselected records are output to a specified file. Unselected records are always in binary format.

The /REJECTED qualifier allows you to specify the name of the file for the unselected records. If you omit the device or directory specification, the current device and default directory are used. If you omit the file name, then the file name of the input file is used. If you omit the file type, REJ is used.

If you specify /NOREJECTED or omit the qualifier, unselected records are not output.

/REMOTE_ID=(["-",]remote-id[,...])
/NOREMOTE_ID

Controls whether only those records matching the specified remote-id are selected. The remote-id identifies the process or user on a remote node. The exact format of a remote-id varies with the context and DECnet implementation. (See the DECnet user's guide for the operating system on the remote node.)

When you specify /REMOTE_ID, you must specify at least one remote-id. If you specify more than one remote-id, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any remote-id in the list.

If you specify /NOREMOTE_ID or omit the qualifier, the remote-id is not used to select records.

/REPORT[=(report-item[,...])]
/NOREPORT

Controls whether a specified item is included in a summary report. One column is generated on the summarization report for each item specified. See the description of the /SUMMARY qualifier.

If you specify more than one report-item, separate them with commas and enclose the list in parentheses.

The columns on the summarization report appear in the same left-to-right sequence as given in the list of report-items.

You can specify any of the following items:

Keyword	Meaning
BUFFERED_IO	Total buffered IOs
DIRECT_IO	Total direct IOs
ELAPSED	Total elapsed time
EXECUTION	Total images executed
FAULTS	Total page faults
GETS	Total VAX-11 RMS gets
PAGE_FILE	Maximum page file usage
PAGE_READS	Total page read IOs
PAGES	Total pages printed
PROCESSOR	Total processor time consumed
QIOS	Total QIOs issued
RECORDS	Total records in file (default)
VOLUMES	Total volumes mounted
WORKING_SET	Maximum working set size

If you specify /REPORT without a value (or you specify /SUMMARY and do not specify /REPORT) then /REPORT=RECORDS is assumed. Many of these report items are present in only a few types of accounting records. If records are selected that do not contain a report value that has been requested, a default value, usually zero, is used.

/SINCE[=time]
/NOSINCE

Controls whether only those records dated later than a specified time are selected. You can specify an absolute time, delta time, or a combination of the two. Observe the syntax rules for date and time described in the VAX/VMS Command Language User's Guide.

If you specify /SINCE without the time, midnight of the current day is used.

If you specify /NOSINCE or omit the qualifier, no time is used to select records.

/SORT[=([-]sort-item[,...])]
/NOSORT

Specifies the sequence of the records in the brief or full listing. By default the sequence is the same as that of the input files.

At least one sort-item must be specified. If you specify more than one sort-item, separate them with commas and enclose the list in parentheses.

If a sort-item is preceded by a minus sign (-), then that field is used as a descending key. By default keys are assumed to be ascending.

The selected records are sorted according to the sequence specified by the sort-items given with the /SORT qualifier prior to writing them to the designated output file. Unselected records are not sorted. The ordering of sort-items in the qualifier value list determines the relative ranking of the keys.

Note that if a sort-item specifies a field that is not present in a record, that record becomes unselected and will be reflected as such in the counts of selected and rejected records.

You can specify any of the following sort-items:

Keyword	Meaning
ACCOUNT	User's account name
ADDRESS	Remote node address
BUFFERED_IO	Buffered IO count
DIRECT_IO	Direct IO count
ELAPSED	Elapsed time
ENTRY	Number of batch or print job queue entry
EXECUTION	Image execution count
FAULTS	Page faults
FINISHED	Termination time or time record was written
GETS	Number of gets from the file to be printed
IDENT	Process identification
IMAGE	Image name
JOB	Name of batch or print job
NODE	Remote node name
OWNER	Owner process identification
PAGES	Number of pages printed
PAGE_FILE	Peak page file usage
PAGE_READS	Page read IOs
PRIORITY	Process base priority
PROCESS	Process type
PROCESSOR	Processor time
QIOS	Number of QIOs to the printer
QUEUE	Name of queue
QUEUED	Time batch or print job was queued
STARTED	Start time
STATUS	Final exit status
TERMINAL	Terminal name
TYPE	Record type
UIC	User identification code
USER	User's name
VOLUMES	Number of volumes mounted
WORKING_SET	Peak working set size

/STATUS=(["-",]exit-status[,...])
/NOSTATUS

Controls whether only those records matching the specified exit-status are selected. The exit-status refers to the final completion status of the process or image.

When you specify /STATUS, you must specify at least one exit-status. If you specify more than one exit-status, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any exit-status in the list.

If you specify /NOSTATUS or omit the qualifier, the exit-status is not used to select records.

/SUMMARY[=(summary-item[,...])]
/NOSUMARY

Specifies that a summary of the selected records, grouped by the list of summary keys, be produced. Use the /REPORT qualifier to control what information is summarized. If you omit the /REPORT qualifier, then /REPORT=RECORDS is assumed.

If you specify /SUMMARY without a value, then /SUMMARY=USER is assumed.

If you specify more than one summary-item, separate them with commas and enclose the list in parentheses.

The summarized items are sorted in ascending order and listed in the same left-to-right sequence given in the list of summary-items. The output is sent to the SYS$OUTPUT device unless specifically directed elsewhere by the /OUTPUT qualifier.

You can specify any of the following summary items:

Keyword	Outputs
ACCOUNT	Account name from the UAF
DATE	YYYY MM DD
DAY	Day of month (1-31)
HOUR	Hour of day (0-23)
IMAGE	Image name
JOB	Name of batch job or print job
MONTH	Month of year (1-12)
NODE	Remote node name
PROCESS	Process type
QUEUE	Batch or device queue name
TERMINAL	Terminal name
TYPE	Type of record (logout, batch)
UIC	User identification code
USER	User name from UAF
WEEKDAY	Day of week (1=Sunday, 2=Monday, and so on)
YEAR	Year

If you specify /NOSUMMARY or omit the qualifier, no summarization occurs.

/TERMINAL=(["-",]terminal-name[,...])
/NOTERMINAL

Controls whether only those records matching the specified terminal-names are selected. Terminal names are associated with interactive processes.

When you specify /TERMINAL, you must specify at least one terminal-name. Specify terminal-names as a standard device names and include the colon (:), for example, TTA6:.

If you specify more than one terminal-name, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any terminal-name in the list.

If you specify /NOTERMINAL or omit the qualifier, the terminal-name is not used to select records.

/TITLE=title
/NOTITLE

Specifies the title to be printed in the center of the first line of summary reports. The title line also includes the beginning and ending times for the data summary at the left and right margins, respectively.

If the title includes spaces or special characters, you must enclose it in quotation marks (").

/TYPE=(["-",]record-type[,...])
/NOTYPE

Controls whether only those records matching the specified record-type are selected.

When you specify /TYPE, you must specify at least one record-type. If you specify more than one record-type, separate them with commas and enclose the list in parentheses.

You can specify any of the following record types:

Keyword	Meaning
FILE	Accounting file forward and backward pointers
IMAGE	Termination of image
LOGFAIL	Unsuccessful conclusion of a login attempt
PRINT	Termination of print job
PROCESS	Termination of process
SYSINIT	System initialization
UNKNOWN	Any record not recognized as one of the above
USER	Arbitrary user messages

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any record-type in the list.

If you specify /NOTYPE or omit the qualifier, the record-type is not used to select records.

/UIC=(["-",]uic[,...])
/NOUIC

Controls whether only those records matching the specified user identification code (UIC) are selected.

When you specify /UIC, you must specify at least one UIC. If you specify more than one UIC, separate them with commas and enclose the list in parentheses.

Specify the UIC in the format:

[g,m]

g is an octal number in the range 0 through 377 representing the group number

m is an octal number in the range 0 through 377 representing the member number

Square brackets ([]) or angle brackets (<>) are required in the UIC specification.

You can specify the asterisk (*) wild card character in either the group or member fields of the UIC specification, or in both.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any UIC in the list.

If you specify /NOUIC or omit the qualifier, the UIC not used to select records.

/USER=(["-",]user-name[,...])
/NOUSER

Controls whether only those records matching the specified user-name are selected. The user-name matches the user name in the user authorization file.

When you specify /USER, you must specify at least one user-name. If you specify more than one user-name, separate them with commas and enclose the list in parentheses.

If the first keyword in the list is a minus sign enclosed in quotation marks ("-"), all records are selected except those matching any user-name in the list.

If you specify /NOUSER or omit the qualifier, the user-name is not used to select records.

1.3 OUTPUTS

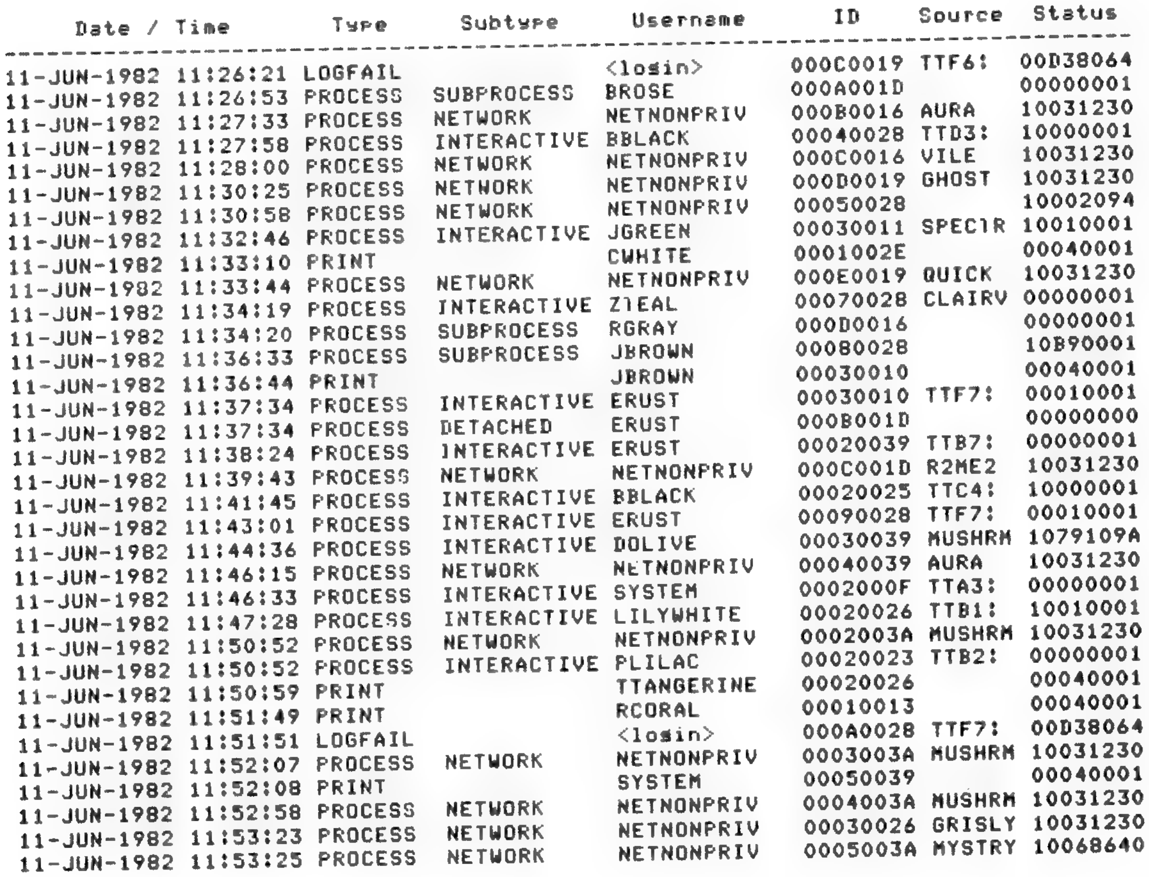
The Accounting Utility can produce combinations of output for any single ACCOUNTING request. The basic forms of output are listing and binary output.

1.3.1 Listing Output

Listing output consists of terminal screen images. Any terminal supported by VAX/VMS with dimensions of at least 80 columns by 24 rows can be used. (You may have to issue the DCL command SET TERMINAL to set the proper dimensions.) Listing output can also be routed to a file for subsequent printing.

There are three basic screen formats used for displaying data: the brief listing, the full listing, and the summarization report. These screen formats are described in the following sections, with a sample of each.

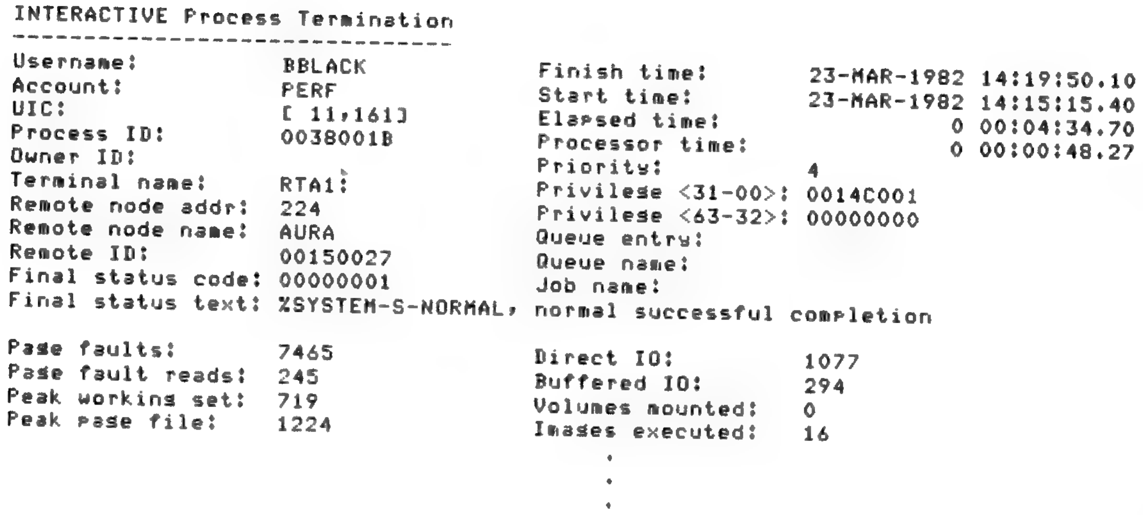
1.3.1.1 **Brief Listing Format** - The brief listing format provides one line for each record in the accounting file being processed. The output always includes the date and time, the type of record, the subtype, the user name, the ID, the source, and the status. See Figure 1-1.



```
      Date / Time        Type     Subtype      Username       ID     Source  Status
--------------------------------------------------------------------------------------
11-JUN-1982 11:26:21 LOGFAIL                 <login>       000C0019 TTF6:   00D38064
11-JUN-1982 11:26:53 PROCESS  SUBPROCESS  BROSE         000A001D         00000001
11-JUN-1982 11:27:33 PROCESS  NETWORK     NETNONPRIV    000B0016 AURA    10031230
11-JUN-1982 11:27:58 PROCESS  INTERACTIVE BBLACK        00040028 TTD3:   10000001
11-JUN-1982 11:28:00 PROCESS  NETWORK     NETNONPRIV    000C0016 VILE    10031230
11-JUN-1982 11:30:25 PROCESS  NETWORK     NETNONPRIV    000D0019 GHOST   10031230
11-JUN-1982 11:30:58 PROCESS  NETWORK     NETNONPRIV    00050028         10002094
11-JUN-1982 11:32:46 PROCESS  INTERACTIVE JGREEN        00030011 SPEC1R  10010001
11-JUN-1982 11:33:10 PRINT                CWHITE        0001002E         00040001
11-JUN-1982 11:33:44 PROCESS  NETWORK     NETNONPRIV    000E0019 QUICK   10031230
11-JUN-1982 11:34:19 PROCESS  INTERACTIVE Z1EAL         00070028 CLAIRV  00000001
11-JUN-1982 11:34:20 PROCESS  SUBPROCESS  RGRAY         000D0016         00000001
11-JUN-1982 11:36:33 PROCESS  SUBPROCESS  JBROWN        00080028         10B90001
11-JUN-1982 11:36:44 PRINT                JBROWN        00030010         00040001
11-JUN-1982 11:37:34 PROCESS  INTERACTIVE ERUST         00030010 TTF7:   00010001
11-JUN-1982 11:37:34 PROCESS  DETACHED    ERUST         000B001D         00000000
11-JUN-1982 11:38:24 PROCESS  INTERACTIVE ERUST         00020039 TTB7:   00000001
11-JUN-1982 11:39:43 PROCESS  NETWORK     NETNONPRIV    000C001D R2ME2   10031230
11-JUN-1982 11:41:45 PROCESS  INTERACTIVE BBLACK        00020025 TTC4:   10000001
11-JUN-1982 11:43:01 PROCESS  INTERACTIVE ERUST         00090028 TTF7:   00010001
11-JUN-1982 11:44:36 PROCESS  INTERACTIVE DOLIVE        00030039 MUSHRM  1079109A
11-JUN-1982 11:46:15 PROCESS  NETWORK     NETNONPRIV    00040039 AURA    10031230
11-JUN-1982 11:46:33 PROCESS  INTERACTIVE SYSTEM        0002000F TTA3:   00000001
11-JUN-1982 11:47:28 PROCESS  INTERACTIVE LILYWHITE     00020026 TTB1:   10010001
11-JUN-1982 11:50:52 PROCESS  NETWORK     NETNONPRIV    0002003A MUSHRM  10031230
11-JUN-1982 11:50:52 PROCESS  INTERACTIVE PLILAC        00020023 TTB2:   00000001
11-JUN-1982 11:50:59 PRINT                TTANGERINE    00020026         00040001
11-JUN-1982 11:51:49 PRINT                RCORAL        00010013         00040001
11-JUN-1982 11:51:51 LOGFAIL              <login>       000A0028 TTF7:   00D38064
11-JUN-1982 11:52:07 PROCESS  NETWORK     NETNONPRIV    0003003A MUSHRM  10031230
11-JUN-1982 11:52:08 PRINT                SYSTEM        00050039         00040001
11-JUN-1982 11:52:58 PROCESS  NETWORK     NETNONPRIV    0004003A MUSHRM  10031230
11-JUN-1982 11:53:23 PROCESS  NETWORK     NETNONPRIV    00030026 GRISLY  10031230
11-JUN-1982 11:53:25 PROCESS  NETWORK     NETNONPRIV    0005003A MYSTRY  10068640
```

Figure 1-1: Sample Brief Listing

1.3.1.2 **Full Listing Format** - The full listing format provides all the data for each record in the accounting file being processed. There are small variations in the record formats, based on the presence or absence of data in the record. Figure 1-2 presents a fairly typical single accounting record in the full format.



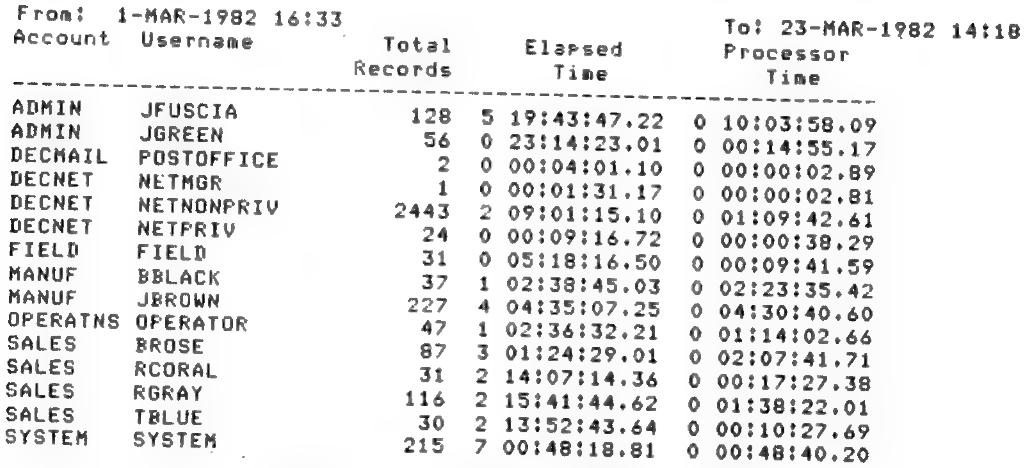
```
INTERACTIVE Process Termination
-------------------------------
Username:           BBLACK             Finish time:        23-MAR-1982 14:19:50.10
Account:            PERF               Start time:         23-MAR-1982 14:15:15.40
UIC:                [ 11,161]          Elapsed time:                 0 00:04:34.70
Process ID:         0038001B           Processor time:               0 00:00:48.27
Owner ID:                              Priority:           4
Terminal name:      RTA1:              Privilege <31-00>:  0014C001
Remote node addr:   224                Privilege <63-32>:  00000000
Remote node name:   AURA               Queue entry:
Remote ID:          00150027           Queue name:
Final status code:  00000001           Job name:
Final status text:  %SYSTEM-S-NORMAL, normal successful completion

Page faults:        7465               Direct IO:          1077
Page fault reads:   245                Buffered IO:        294
Peak working set:   719                Volumes mounted:    0
Peak page file:     1224               Images executed:    16
                                          .
                                          .
                                          .
```

Figure 1-2: Sample Full Listing

1.3.1.3 **Summary Output** - Summary output is an ASCII file consisting of the specified report-item values grouped by the specified summary-items. The summary file reflects the accumulation throughout the ACCOUNTING period requested. The statistics in the summary output are either totals or maximum values encountered. (See the /REPORT qualifier for details.)

The summary output format includes the dates of the accounting period on the first line. The beginning date appears at the left, optionally followed by the title specified with the /TITLE qualifier. The ending date for the report appears at the right margin.

Figure 1-3 illustrates a summary output.



```
From:  1-MAR-1982 16:33                                    To: 23-MAR-1982 14:18
Account   Username              Total       Elapsed          Processor
                               Records        Time              Time
-------------------------------------------------------------------------
ADMIN     JFUSCIA                  128  5 19:43:47.22   0 10:03:58.09
ADMIN     JGREEN                    56  0 23:14:23.01   0 00:14:55.17
DECMAIL   POSTOFFICE                 2  0 00:04:01.10   0 00:00:02.89
DECNET    NETMGR                     1  0 00:01:31.17   0 00:00:02.81
DECNET    NETNONPRIV              2443  2 09:01:15.10   0 01:09:42.61
DECNET    NETPRIV                   24  0 00:09:16.72   0 00:00:38.29
FIELD     FIELD                     31  0 05:18:16.50   0 00:09:41.59
MANUF     BBLACK                    37  1 02:38:45.03   0 02:23:35.42
MANUF     JBROWN                   227  4 04:35:07.25   0 04:30:40.60
OPERATNS  OPERATOR                  47  1 02:36:32.21   0 01:14:02.66
SALES     BROSE                     87  3 01:24:29.01   0 02:07:41.71
SALES     RCORAL                    31  2 14:07:14.36   0 00:17:27.38
SALES     RGRAY                    116  2 15:41:44.62   0 01:38:22.01
SALES     TBLUE                     30  2 13:52:43.64   0 00:10:27.69
SYSTEM    SYSTEM                   215  7 00:48:18.81   0 00:48:40.20
```

Figure 1-3: Sample Summary Output

1.3.2 Binary Output Files

A binary output file is a VAX-11 RMS sequential binary file that is created when an ACCOUNTING request includes the /BINARY or /REJECTED qualifier. All resulting files can be used as source files by later requests to format and display the data, to create a summary file, or to record a new binary file with different selection criteria.

With the /BINARY qualifier, all data pertaining to the request is recorded.

With the /REJECTED qualifier, only those records not being selected are directed to this file.

A complete description of the ACCOUNTING record formats appears in Appendix C.

1.4 BASIC MODES OF OPERATION

This section describes some of the ways you can use ACCOUNTING at your site.

1.4.1 Listing Accounting Files

Use this mode when you want to examine the activity of the system, either on a routine basis, or as part of an installation checkout, tuning, or trouble-shooting exercise. No historical record of output is kept.

The following examples illustrate the listing mode of operation.

```
     $ ACCOUNTING
```

This command produces a display of all accounting records. Since no command qualifiers have been named, ACCOUNTING applies the following defaults to the display:

```
     /NOFULL = a brief listing
     input file = SYS$MANAGER:ACCOUNTNG.DAT
     /OUTPUT = current SYS$OUTPUT device
```

By default, listing begins when the command is issued and ends when you reach the end of the file.

The next command also provides a brief listing, but lists from a specified binary input file.

```
     $ ACCOUNTING MYFILE
```

The next command provides a full listing of all the records in the current accounting file:

```
     $ ACCOUNTING/FULL
```

1.4.1.1 Selecting Records

- You identify groups of accounting records with one or more of the following selection qualifiers: /ACCOUNT, /ADDRESS, /BEFORE, /ENTRY, /IDENTIFICATION, /IMAGE, /JOB, /NODE, /OWNER, /PRIORITY, /PROCESS, /QUEUE, /REMOTE_ID, /SINCE, /STATUS,

/TERMINAL, /TYPE, /UIC, and /USER. If you omit these qualifiers, the defaults provide for selecting all records. The next example illustrates selection.

```
     $ ACCOUNTING /SINCE=22-JUN-1982:00:00:00 -
     $_/BEFORE=22-JUN-1982:23:59:99/ACCOUNT=MANUFA/NODE=OSCAR -
     $_/TYPE=LOGFAIL
```

This command selects and lists in brief format only those records from the node OSCAR and its manufacturing account that represent login failures on a particular day.

Selection is also valuable for summary reports, as the following example illustrates:

```
     $ ACCOUNTING/TYPE=PROCESS/PROCESS=INTERACTIVE -
     $_/PRIORITY=("-",4)/SUMMARY=(USER,TERMINAL) -
     $_/REPORT=(RECORDS,ELAPSED,PROCESSOR)
```

This command selects all interactive process terminations that had a base priority other than 4, summarizes them by user name and terminal name, and reports the number of records of each, their total elapsed time, and the total processor time. The output is directed to the current SYS$OUTPUT device.

1.4.1.2 Sorting Records - The Accounting Utility includes a sorting facility. You use the /SORT qualifier to specify the fields that you want the sort to occur on and whether the desired sequence is ascending or descending. For example:

```
     $ ACCOUNTING/SORT=(USER,ACCOUNT,-STATUS)
```

This command sorts all records in the current accounting file by first the user name and then the account in ascending order and by final status in descending order to provide a brief listing.

You can also sort just those records that you select, by combining one or more selection qualifiers with the /SORT qualifier. However, you only use the /SORT qualifier for brief or full listings, not for summary reports. For example:

```
     $ ACCOUNTING/TYPE=PRINT/QUEUE=LPC0/SORT=USER
```

This command selects all print jobs completed on queue LPC0 and then sorts them by user name, producing a brief listing.

1.4.2 Directing the ACCOUNTING Output

ACCOUNTING output can be routed to any supported terminal device or to a disk or tape file. The following command sends its ASCII listing of all records in the accounting file to the file ACCOPY.LIS. This file could then be printed out on a hard-copy device.

```
     $ ACCOUNTING /OUTPUT=ACCOPY
```

You can also direct binary output to a file whenever you need to capture ACCOUNTING data for future use. For example, you might want to plan for routine performance data gathering for long-term analysis. ACCOUNTING data can be recorded on a routine basis and summarized to gather data about system resource utilization over long periods of time.

The following example illustrates another case of directing the output to a separate file.

```
$ ACCOUNTING/TYPE=PROCESS/PROCESS=BATCH/QUEUE=("-",SYS$BATCH) -
$_/OUTPUT=BATCH/REJECTED=NOBATCH
```

This command uses the current accounting file and selects all records that are batch process terminations for all queues other than SYS$BATCH. The selected records are formatted into a brief listing and output to a BATCH.LIS. The rejected records are copied in their binary form to NOBATCH.REJ.

1.5 USAGE CONSIDERATIONS

This section contains information to help you run the utility effectively.

1.5.1 DCL Symbols

It may be convenient to establish DCL symbols for frequently used combinations, as in the example below.

```
$ MY_GROUP :== "/USER=(MARY,TOM,DICK,HARRY,BARNEY,ALICE) -
$_/PROCESS=INTERACTIVE/TYPE=PROCESS"
$_ACCOUNTING 'MY_GROUP'
```

1.5.2 File Space

When output file recording is active, it is possible to consume large quantities of disk space in a short period of time. In particular, if disk quota is exceeded during execution of an ACCOUNTING request, open files are closed and the request is terminated prematurely. To avoid this situation, carefully plan recording requests by estimating the amount of disk space required.

If necessary, refer to Appendix C, Format of Accounting Log File Data, for details on the exact record sizes for this version of VAX/VMS ACCOUNTING.

1.5.3 Processing Time Can Be Significant

The size of the file being processed and the type of processing being done (for example, sorting) can require significant processing time, which may be particularly noticeable on heavily loaded systems. If this becomes a problem, you may want to run accounting jobs in batch mode.

1.6 ERROR MESSAGES

The VAX/VMS System Messages and Recovery Procedures Manual lists the messages generated by ACCOUNTING and provides explanations and suggested user actions.

CHAPTER 2

AUTHORIZE UTILITY (AUTHORIZE)

The Authorize Utility (AUTHORIZE) is a system management tool that modifies the existing user authorization file (UAF) or creates a new UAF. The system user authorization file (SYS$SYSTEM:SYSUAF.DAT) is described in the VAX/VMS System Management and Operations Guide.

2.1 INVOKING AND TERMINATING AUTHORIZE

The following commands invoke the utility:

```
     $ SET DEFAULT SYS$SYSTEM
     $ RUN AUTHORIZE
```

Your default device and directory must be that of SYS$SYSTEM to access the system UAF.

You can create and maintain UAF files in other directories, using AUTHORIZE. Such files, however, have no effect on system operation. It might be useful to prepare and save one or more authorization files to match special conditions you expect to occur at varying times. Then, whenever necessary, you could use the file to temporarily replace the current SYSUAF.DAT. However, before you go to this effort you should carefully consider whether or not you could use the designations of primary and secondary days and hours to accomplish the same thing.

If a system UAF exists, the system responds with the following prompt:

```
     UAF>
```

You can then enter any of the commands listed in Table 2-1.

If no system UAF exists (that is, you have deleted it), the system issues an error message and the following prompt:

```
     Do you want to create a new file?
```

A response of YES (or Y) results in creation of a new system UAF containing a DEFAULT record and a SYSTEM record. These records are initialized with the values shown in Tables 2-2 and 2-3.

Use of the Authorize Utility requires write access to SYS$SYSTEM:SYSUAF.DAT. Write access to the file is normally restricted to users with a system UIC or the SYSPRV privilege. You can further restrict access to the UAF with the DCL command SET PROTECTION. However, you should not expand access rights to the UAF.

Because certain images (such as MAIL and SET) require access to the system UAF, and are normally installed with the SYSPRV privilege, make certain you always grant system access to SYSUAF.DAT.

The authorization file is created with the following default protection:

```
     SYSUAF.DAT -- (S:RWED,O:RWED,G,W)
```

If you need to maximize the protection for SYSUAF.DAT, the following DCL command is recommended:

```
     $ SET PROTECTION=(S:RWED,O,G,W) SYS$SYSTEM:SYSUAF.DAT
```

You can terminate AUTHORIZE with the EXIT command or by pressing CTRL/Z.

2.2 AUTHORIZE COMMANDS

Table 2-1 summarizes the AUTHORIZE commands. The ADD, COPY, DEFAULT, and MODIFY commands act upon individual fields of system UAF records through the specification of appropriate qualifiers. Table 2-2 lists the qualifiers, describes the corresponding fields, and specifies the defaults (as provided in the DEFAULT record in the software distribution kit). Table 2-3 specifies the values of the qualifiers in the SYSTEM, FIELD, and SYSTEST records (as provided with the software distribution kit).

Table 2-1: AUTHORIZE Command Summary

Format	Function
ADD user-name [/qualifier[...]]	Adds a system UAF record
COPY old-user-name new-user-name [/qualifier[...]]	Copies a system UAF record
DEFAULT /qualifier[...]	Modifies the DEFAULT system UAF record
EXIT	Returns the user to DCL command level
HELP [command-name [/qualifier]]	Displays the AUTHORIZE commands
LIST [user-spec] [/FULL]	Creates a listing file of system UAF records
MODIFY user-name /qualifier[...]	Modifies one or more system UAF records
REMOVE user-name	Deletes a system UAF record
RENAME old-user-name new-user-name [/PASSWORD[=password]]	Renames a system UAF record
SHOW user-spec [/BRIEF]	Displays system UAF records

Table 2-2: System UAF Field Qualifiers

Qualifier	Default	Description
/ACCOUNT=account-name	blanks	Default account name (for example, a billing name or number) -- 1 through 8 characters
/ASTLM=value	10	AST queue limit (total asynchronous system trap (AST) operations and scheduled wake-up requests that can be outstanding at one time) -- integer; should be minimum of 2
/BIOLM=value	6	Buffered I/O count limit (total of buffered I/O operations, such as terminal I/O, that can be outstanding at one time) -- integer; should be minimum of 2
/BYTLM=value	4096	Buffered I/O byte limit (total bytes that can be specified for transfer in outstanding buffered I/O operations) -- integer; should be minimum of 1024
/CLI=cli-name	DCL	Name of the default command interpreter -- 1 through 9 characters; should be DCL or MCR
/CPUTIME=delta-time	0	Maximum CPU time that a user's process can take per session -- unit of time in delta format; 0 means infinite time[1]
/DEVICE=device-name	blanks	Name of default device (must be a direct-access device) -- 1 through 15 alphanumeric characters; colon is appended if omitted; at login time, a blank value is interpreted as equivalent to SYS$SYSDEVICE

(continued on next page)

[1] Indicates that this value can be overridden with the corresponding DCL qualifier on the INITIALIZE/QUEUE, SET/QUEUE/ENTRY, or SUBMIT commands. See the VAX/VMS Command Language User's Guide for more information.

Table 2-2 (Cont.): System UAF Field Qualifiers

Qualifier	Default	Description
/DIOLM=value	6	Direct I/O count limit (total direct (usually disk) I/O operations that can be outstanding at one time) -- integer; should be minimum of 2
/DIRECTORY= directory-name	[USER]	Default directory name -- 1 through 31 characters; brackets are provided if omitted
/ENQLM=value	10	Limit for the number of locks that can be queued at one time; minimum is 2
/FILLM=value	20	Open file limit (total files that can be open at one time, including active network logical links) -- integer; should be minimum of 2
/FLAGS=	(see below)	Login flags -- NO in front of a flag name clears the flag; parentheses may be omitted for a single flag name; multiple flag names must be separated by commas
([[NO]CAPTIVE]	NOCAPTIVE	Restricts the user in several areas: disables CTRL/Y interrupts, prohibits use of the SET PASSWORD command, and prohibits user specification of a default CLI. Also, the user may not specify /DISK or /COMMAND when logging in. This flag is typically used to prevent an application's user from having unrestricted access to the CLI
[,[NO]DEFCLI]	NODEFCLI	Restricts the user to using the default command interpreter; prohibits use of the /CLI qualifier at login time (however, the MCR command can still be used)

(continued on next page)

Table 2-2 (Cont.): System UAF Field Qualifiers

Qualifier	Default	Description
[,[NO]DISCTLY]	NODISCTLY	Disables future CTRL/Y interrupts. If the intent of DISCTLY is only to force execution of the login command procedure, that procedure should issue a SET CONTROL_Y command before exiting
[,[NO]DISUSER]	NODISUSER	Disables this user from logging in
[,[NO]DISNEWMAIL]	NODISNEWMAIL	Suppresses announcements of new mail at login time
[,[NO]DISWELCOME]	NODISWELCOME	Suppresses the login message "Welcome to ..."
[,[NO]LOCKPWD])	NOLOCKPWD	Locks the user's password; prohibits use of the SET PASSWORD command
/LGICMD=file-spec	blanks	Name of login command file -- the standard file specification with the following defaults: device is as specified for /DEVICE, directory is as specified for /DIRECTORY, file type is COM (if command interpreter is DCL) or CMD (if command interpreter is MCR)
/MAXACCTJOBS	0	Unimplemented
/MAXJOBS=value	0	Maximum number of active processes (interactive, batch, and detached) permitted at one time for this user; the default value of 0 represents an unlimited number
/OWNER=owner-name	blanks	Name of owner (for billing purposes, for example) -- 1 through 32 characters
/PASSWORD[=password]	USER	Password for logging in -- 0 through 31 characters. Passwords, if specified, can only consist of alphanumeric characters, dollar signs, or underscores. If a null password is specified, the

(continued on next page)

Table 2-2 (Cont.): System UAF Field Qualifiers

Qualifier	Default	Description
/PASSWORD[=password] (Cont.)		user is not prompted for a password at login time. Specify a null password with the /PASSWORD qualifier without an equal sign and a password string
/PBYTLM=value	0	Unimplemented
/PFLAGS=	(see below)	Login flags for primary days -- NO in front of a flag clears the flag; parentheses may be omitted for a single flag name; multiple flag names must be separated by commas
([[NO]DISDIALUP]	NODISDIALUP	Restricts the user from logging in on a dial-up line during a primary day
[,[NO]DISNETWORK])	NODISNETWORK	Restricts the user from logging in with the DCL command SET HOST during a primary day. This does not, however, prevent the user from using SET HOST once logged in
/PGFLQUOTA=value	10000	Paging file limit (total pages that a user process can use in the system paging file) -- integer; should be minimum of 2048 for typical interactive processes
/PRCLM=value	2	Subprocess creation limit (total subprocesses that can exist at one time) -- integer; minimum of 0
/P_RESTRICT [=(range[,...])]	no hourly restrictions	Specifies one or more ranges of hours when logins are prohibited on primary days. By default, there are no hourly restrictions. Specify the hours as integers between 0 and 23, inclusive. You can specify single hours or groups of hours. To specify a range of hours, separate the beginning and ending hour with a hyphen (-). (You can specify

(continued on next page)

Table 2-2 (Cont.): System UAF Field Qualifiers

Qualifier	Default	Description
/P_RESTRICT (Cont.)		hours that wrap around the clock, such as the range 22-6. In this example, logins would be prohibited between 0:00 and 6:59 and between 22:00 and 23:59 on any primary day.) The user can log in during any hour that is not specified with the qualifier. Furthermore, each new specification of the qualifier completely resets all existing hourly restrictions. Thus, you can remove all previous restrictions by specifying /P_RESTRICT without an equal sign and range of hours. Parentheses can be omitted for a single hour or range of hours. Multiple ranges must be separated by commas
/PRIMEDAYS= ([NO]day[,...])	MON, TUE, WED, THU, FRI, NOSAT, NOSUN	Defines the primary and secondary days of the week for login purposes. If you omit this qualifier, by default the primary days are Monday through Friday and the secondary days are Saturday and Sunday. To designate a day as a secondary day, use the prefix NO. Unique abbreviations are accepted. Any days omitted from the list take their default value
/PRIO=value	4	Default base priority for user processes -- integer; should be in the range of 0 through 31; 4 is the norm for timesharing users
/PRIVILEGES= ([NO]priv-name[,...])	NETMBX TMPMBX	Privileges allotted user -- names of privileges; when using MODIFY, the specified privileges are added to the existing ones; NO in front of a priv-name removes the privilege; parentheses may be omitted for a

(continued on next page)

Table 2-2 (Cont.): System UAF Field Qualifiers

Qualifier	Default	Description
/PRIVILEGES= (Cont.)		single priv-name; multiple priv-names must be be separated by commas
/SFLAGS=	(see below)	Login flags for secondary days -- NO in front of a flag clears the flag; parentheses may be omitted for a single flag name; multiple flag names must be separated by commas
([[NO]DISDIALUP]	NODISDIALUP	Restricts the user from logging in on a dial-up line during a secondary day
[,[NO]DISNETWORK])	NODISNETWORK	Restricts the user from logging in with the DCL command SET HOST during a secondary day. This does not, however, prevent the user from using SET HOST once logged in
/S_RESTRICT [=(range[,...])]	no hourly restrictions	Specifies one or more ranges of hours when logins are prohibited on a secondary day. By default, there are no hourly restrictions. Specify the hours as integers between 0 and 23, inclusive. You can specify single hours or groups of hours. To specify a range of hours, separate the beginning and ending hour with a hyphen (-). (You can specify hours that wrap around the clock, such as the range 22-6. In this example, logins would be prohibited between 0:00 and 6:59 and between 22:00 and 23:59 on any secondary day.) The user can log in during any hour that is not specified with the qualifier. Furthermore, each new specification of the qualifier completely resets all existing hourly restrictions. Thus, you can remove all previous restrictions by specifying

(continued on next page)

Table 2-2 (Cont.): System UAF Field Qualifiers

Qualifier	Default	Description
/S_RESTRICT (Cont.)		/S_RESTRICT without an equal sign and range of hours. Parentheses can be omitted for a single hour or range of hours. Multiple ranges must be separated by commas
/SHRFILLM=value	0	Unimplemented
/TQELM=value	10	Timer queue entry limit (total entries in the timer queue plus the number of temporary common event flag clusters that the user's process can have at one time) -- integer; minimum of 0
/UIC=uic	[200,200]	User identification code; group number and member number separated by a comma and enclosed in brackets; each number must be octal in the range 0 through 377
/WSDEFAULT=value	150	Default working set size in physical pages for a user process -- integer; should be minimum of 50. This is also the default size that a user process is set to after the completion of any image. The user is allowed to alter this quantity up to WSQUOTA with the DCL command SET WORKING_SET. A value of 150 is satisfactory for most applications
/WSEXTENT=value	500	Working set extent -- this value represents the absolute limit on physical memory that the system will allow this process to have. The memory over and above WSQUOTA is only available to this process when the system has an excess of free pages. This additional memory will be taken back by the system if needed. Values in the range 500 and up

(continued on next page)

Table 2-2 (Cont.): System UAF Field Qualifiers

Qualifier	Default	Description
/WSEXTENT=value (Cont.)		are typical. The minimum should be greater than or equal to the working set quota[1]
/WSQUOTA=value	200	Limit for the amount of physical memory a user process may lock into its working set. Also an upper limit on the amount of swap space the system will reserve for this process and the upper limit on physical memory that the system will allow this process to consume if the system-wide memory demand is significant -- integer; should be minimum of 50. Values in the range 200-400 are suitable for most applications[1]

[1] Indicates that this value can be overridden with the corresponding DCL qualifier on the INITIALIZE/QUEUE, SET/QUEUE/ENTRY, or SUBMIT commands. See the VAX/VMS Command Language User's Guide for more information.

Table 2-3: Initial Values of SYSTEM, FIELD, and SYSTEST Records

Qualifier	SYSTEM	FIELD	SYSTEST
/ACCOUNT	SYSTEM	FIELD	SYSTEST
/ASTLM	20	10	10
/BIOLM	12	12	12
/BYTLM	20480	10240	20480
/CLI	DCL	DCL	DCL
/CPUTIME	0	0	0
/DEVICE	blanks	blanks	blanks
/DIOLM	12	12	12
/DIRECTORY	[SYSMGR]	[SYSMAINT]	[SYSTEST]

(continued on next page)

Table 2-3 (Cont.): Initial Values of SYSTEM, FIELD, and SYSTEST Records

Qualifier	SYSTEM	FIELD	SYSTEST
/ENQLM	20	10	20
/FILLM	20	20	20
/FLAGS	no restrictions	no restrictions	no restrictions
/LGICMD	blanks	blanks	blanks
/MAXACCTJOBS	0	0	0
/MAXJOBS	0	0	0
/OWNER	SYSTEM MANAGER	FIELD SERVICE	SYSTEST-UETP
/PASSWORD	MANAGER	SERVICE	UETP
/PBYTLM	0	0	0
/PFLAGS	NODISDIALUP, NODISNETWORK	NODISDIALUP, NODISNETWORK	NODISDIALUP, NODISNETWORK
/PGFLQUOTA	10000	10000	10000
/PRCLM	10	2	8
/P_RESTRICT	no restrictions	no restrictions	no restrictions
/PRIMEDAYS	MON,TUE,WED, THU,FRI	MON,TUE,WED, THU,FRI	MON,TUE,WED, THU,FRI
/PRIO	4	4	4
/PRIVILEGES	all	GRPNAM ALLSPOOL GROUP DIAGNOSE PRMCEB LOG_IO SETPRV TMPMBX PRMMBX PHY_IO NETMBX	CMKRNL CMEXEC SYSNAM GRPNAM DETACH LOG_IO GROUP PRMCEB PRMMBX TMPMBX NETMBX VOLPRO SYSPRV PHY_IO DIAGNOSE
/SFLAGS	NODISDIALUP, NODISNETWORK	NODISDIALUP, NODISNETWORK	NODISDIALUP, NODISNETWORK
/SHRFILLM	0	0	0
/S_RESTRICT	no restrictions	no restrictions	no restrictions
/TQELM	20	20	20
/UIC	[001,004]	[001,010]	[001,007]
/WSDEFAULT	150	150	150
/WSEXTENT	1024	1024	1024
/WSQUOTA	1024	1024	1024

The following sections present the AUTHORIZE commands in alphabetical order. The commands follow the standard rules of DCL grammar, as specified in the *VAX/VMS Command Language User's Guide*. Thus, you can abbreviate any command, keyword, or qualifier as long as the abbreviation is not ambiguous.

2.2.1 ADD Command

The ADD command adds a user record to the system UAF. It has the following format:

```
ADD user-name [/qualifier[...]]
```

user-name
: A string of 1 through 12 alphanumeric characters and underscores (_). Although dollar signs ($) are permitted, they are usually reserved for system names.

/qualifier
: A qualifier as listed in Table 2-2.

Qualifiers not specified take their values from the DEFAULT record except that the default password is always USER (no matter what the password in the DEFAULT record is). Typically, you take defaults on the limits, priority, privileges, command interpreter, and sometimes device; as a result, you type only the password, UIC, directory, owner, account, and sometimes device. The following example illustrates a typical ADD command:

```
UAF>ADD ROBIN /PASSWORD=SP0152/UIC=[014,006]-
_/DEVICE=SYS$USER/DIRECTORY=[ROBIN]-
_/OWNER="JOSEPH ROBIN" /ACCOUNT=VMS
user record successfully added
```

Whenever you add a record to the UAF you should also create a first-level directory for the new user, specifying the device name, directory name, and UIC of the UAF record. Protection for the "ordinary" user is normally read, write, execute, and delete access for the system and owner processes, read and execute access for group processes, and no access for world processes. The following DCL command creates a first-level directory for user ROBIN:

```
$ CREATE/DIRECTORY SYS$USER:[ROBIN] /OWNER_UIC=[014,006]
```

The next example creates a user record with a number of restrictions. The record that results from this command, and an explanation of the restrictions it imposes, appear with Figure 2-3, in the description of the SHOW command.

```
UAF>ADD WELCH /PASSWORD=ROB/UIC=[011,051] -
_/DEVICE=SYS$USER/DIRECTORY=[WELCH]/OWNER="ROB WELCH"
_/ACCOUNT=VMS/FLAGS=(CAPTIVE,DISNEWMAIL,DISWELCOME)
_/PFLAGS=DISNETWORK/SFLAGS=(DISNETWORK,DISDIALUP) -
_/LGICMD=SECUREIN/P RESTRICT=(0-2)/S_RESTRICT=(19-6)
user record successfully added
```

2.2.2 COPY Command

The COPY command creates a new system UAF record that duplicates an existing system UAF record. It has the following format:

```
COPY old-user-name new-user-name [/qualifier[...]]
```

old-user-name
User name for an existing user record.

new-user-name
User name (a string of 1 through 12 alphanumeric characters and underscores) for the new user record.

/qualifier
A qualifier as specified in Table 2-2.

Qualifiers not specified in the command remain unchanged. However, since password verification includes the user name as well as the password, it will generally fail when you attempt to use a new user name with an old password. (Only null passwords can be effectively transferred from one user record to another by the COPY command.) Thus, you will probably want to make it a practice to include the password whenever you use the COPY command.

You could add a new record for a new user named Thomas Sparrow that would be identical to that of Joseph Robin (but presumably different from the default record) by issuing the following command:

```
UAF>COPY  ROBIN  SPARROW  /PASSWORD=SP0152
user record copied
```

However, if you wanted to add a record for Thomas Sparrow that was essentially the same as Joseph Robin's but differed in the UIC, directory name, password, and owner, you could use the following command:

```
UAF>COPY ROBIN  SPARROW /UIC=[200,13]/DIRECTORY=[SPARROW] -
_/PASSWORD=THOMAS/OWNER="THOMAS SPARROW"
user record copied
```

2.2.3 DEFAULT Command

The DEFAULT command modifies the system UAF's DEFAULT record. The command has the following format:

```
DEFAULT /qualifier[...]
```

/qualifier
A qualifier as listed in Table 2-2.

Qualifiers not specified in the command remain unchanged.

You typically modify the DEFAULT record when qualifiers normally assigned to a new user differ from the DIGITAL-supplied values. The qualifiers most often needing modification are as follows:

- /CLI -- If the command interpreter is MCR
- /DEVICE -- If most users have the same default device
- /LGICMD -- Where automation of initial housekeeping chores at login time is desired through a specific login command procedure
- /PRIVILEGES -- Where users are normally given different privileges than the DIGITAL-supplied ones

VAX/VMS automates the execution of login command procedures in the following way. First it checks whether or not SYS$SYLOGIN has been defined. If it has, the logical name is translated (in most cases to SYLOGIN.COM) and that procedure is executed. (It can even call other login command procedures.) However, when it completes, there is another check made. If the user's LGICMD field in the UAF specifies a command procedure, that procedure is executed. If LGICMD is blank, the user's file LOGIN.COM is executed automatically, it it exists and the command interpreter is DCL. (Of course, in this case, all users must name their login command files LOGIN.COM.) If the command interpreter is MCR, the user's file LOGIN.CMD is executed automatically.

Thus, the login protocol generally consists of a system-wide login command procedure followed by a user-specific command procedure.

The following example illustrates the DEFAULT command:

```
     UAF>DEFAULT /DEVICE=SYS$USER/LGICMD=SYS$MANAGER:SECURELGN -
     _/PRIVILEGES=(TMPMBX,GRPNAM,GROUP)
```

After modifying the default value record, you should pencil in the changes in the middle column of Table 2-2.

2.2.4 EXIT Command

The EXIT command returns you to DCL command level. It has the following format:

```
     EXIT
```

You can also return to DCL command level by pressing CTRL/Z.

2.2.5 HELP Command

The HELP command lists and explains the AUTHORIZE commands. It has the following format:

```
     HELP [command-name[/qualifier]]
```

command-name
Name of an AUTHORIZE command; see Table 2-1.

/qualifier
Name of an AUTHORIZE qualifier; see Table 2-2.

If you do not specify a command name, HELP displays general information on the commands for which help is available. It then prompts with "Topic?". You can supply a command name or press RETURN. Specification of command names and qualifiers obtains more detailed information. If you respond with RETURN, the HELP command exits. You can also exit from the HELP command by pressing CTRL/Z. Do not respond with CTRL/Y unless you want to exit from both HELP and AUTHORIZE.

If the command you seek help on accepts qualifiers, the display of the help information on the command is followed by the prompt "Subtopic?". You can respond to this prompt with a qualifier name or press RETURN. If you respond with RETURN, HELP prompts with "Topic?". If you want to exit the HELP command directly from this level, press CTRL/Z.

2.2.6 LIST Command

The LIST command creates a listing file of reports for selected system UAF records. It has the following format:

```
LIST [user-spec] [/FULL]
```

user-spec
A user name or a UIC; defaults to list all users. The asterisk (*) and percent sign (%) wild card characters are permitted in the user name.

/FULL
Request for a full report; omission results in a brief report.

You can print the listing file, named SYSUAF.LIS, with the DCL command PRINT.

Specification of a user name results in a single-report. The following example lists a full report for user ROBIN:

```
UAF>LIST ROBIN /FULL
writing listing file
listing file SYSUAF.LIS complete
```

Specification of the asterisk (*) wild card character results in reports for all users in ascending sequence by user name:

```
UAF>LIST *
```

Note, however, that this is the same result you would produce with the following form of the command:

```
UAF>LIST
```

Specification of a UIC results in reports for all users with that UIC. (Normally each user has a unique UIC, but if users share a UIC, the report will show all.) The asterisk wild card character (*) can be used in specifying the UIC, as illustrated in the following examples:

Command	Description
LIST [014,006] /FULL	Lists a full report for the user (or users) with member number 006 in group 014
LIST [014,*]	Lists a brief report for all users in group 014, in ascending sequence by member number
LIST [*,006]	Lists a brief report for all users with a member number of 006
LIST [*,*]	Lists a brief report for all users, in ascending sequence by UIC

Although you are encouraged to provide separate UICs for each user, the LIST command reports users with the same UIC in the order that they were added to the UAF. Full reports include the details of the limits, privileges, login flags, and command interpreter. Brief reports do not include the limits, login flags, and command interpreter, and summarize the privileges. The password is never listed. See the SHOW command for examples.

2.2.7 MODIFY Command

The MODIFY command changes values in a system UAF user record. It has the following format:

```
     MODIFY user-name /qualifier[...]
```

user-name
: Name of a user in the system UAF. The asterisk (*) and percent sign (%) wild card characters are permitted in the user name. When you specify a single asterisk for the user name, you modify the records of all users.

/qualifier
: A qualifier as listed in Table 2-2.

Values not specified in the command remain unchanged. The following example changes the password for user ROBIN:

```
     UAF>MODIFY ROBIN /PASSWORD=SP0172
     user record(s) updated
```

Modifications to system UAF records do not affect users logged in. The modifications take effect the next time the user logs in.

2.2.8 REMOVE Command

The REMOVE command deletes a system UAF user record. It has the following format:

```
     REMOVE user-name
```

user-name
: Name of a user in the system UAF.

The DEFAULT and SYSTEM records cannot be deleted. The following example deletes user ROBIN:

```
     UAF>REMOVE  ROBIN
     record removed from SYSUAF.DAT
```

2.2.9 RENAME Command

The RENAME command renames a system UAF record. It has the following format:

```
     RENAME  old-user-name  new-user-name  [/PASSWORD[=password]]
```

old-user-name
: Name of user currently in the system UAF.

new-user-name
: Changed name desired by the user.

password
: Password specification for this user. See the description of the /PASSWORD qualifier in Table 2-2.

The new user name must adhere to the user name conventions. That is, it can consist of 1 through 12 alphanumeric characters and underscores (_). Although dollar signs ($) are permitted, they are usually reserved for system names.

The RENAME command changes only the user name of the system UAF record and retains the characteristics of the old record. This can be particularly helpful whenever a user's name changes, as perhaps in the case of marriage or divorce. However, since password verification includes the user name as well as the password, it will generally fail when you attempt to use a new user name with an old password. (Only null passwords can be effectively transferred from one user record to another by the RENAME command.) Thus, you will probably want to make it a practice to include the password whenever you use the RENAME command.

The following example shows how you could change the name of user Maryann Hawkes to Maryann Kramerdove.

```
     UAF>RENAME  HAWKES  KRAMERDOVE  /PASSWORD=MARANNKRA
     user record renamed
```

If you omit the /PASSWORD qualifier, you receive a warning message reminding you that it must be changed.

2.2.10 SHOW Command

The SHOW command displays reports for selected UAF records. It has the following format:

```
     SHOW user-spec [/BRIEF]
```

user-spec
: A user name or UIC. The asterisk (*) and percent sign (%) wild card characters are permitted in the user name.

/BRIEF
: Requests a brief report; omission results in a full report.

Specification of a user name results in a single-user report. The following SHOW command produces a full report for user ROBIN:

```
     UAF>SHOW ROBIN
```

Specification of an asterisk wild card character results in reports for all users in ascending sequence by user name:

```
     UAF>SHOW * /BRIEF
```

Specification of a UIC results in reports for all users with the UIC. The asterisk wild card character (*) can be used in specifying the UIC, as illustrated in the following examples:

Command	Description
SHOW [014,006]	Displays a full report for the user (or users) with member number 006 in group 014
SHOW [014,*] /BRIEF	Displays a brief report for all users in group 014, in ascending sequence by member number
SHOW [*,006] /BRIEF	Displays a brief report for all users with a member number of 006
SHOW [*,*] /BRIEF	Displays a brief report for all users, in ascending sequence by UIC

Users with the same UIC are listed in the order that they were added to the system UAF. Full reports include the details of the limits, privileges, login flags, and command interpreter. Brief reports do not include the limits, login flags, and command interpreter, and summarize the privileges. The password is never listed.

Figure 2-1 illustrates the display for user ROBIN. This example corresponds to the example in Section 2.2.1. Note that most defaults are in effect.

```
UAF>SHOW ROBIN

Username:  ROBIN          Owner:  JOSEPH ROBIN
Account:   VMS            UIC:    [014,006]
CLI:       DCL            LGICMD:
Default Device  SYS$USER:
Default Directory: [ROBIN]
Login Flags:
Primary days:   Mon Tue Wed Thu Fri
Secondary days:                     Sat Sun
No hourly restrictions
PRIO:      4  BYTLM:         4096  BIOLM:           6
PRCLM:     2  PBYTLM:           0  DIOLM:           6
ASTLM:    10  WSDEFAULT:      150  FILLM:          20
ENQLM:    10  WSQUOTA:        200  SHRFILLM:        0
TQELM:    10  WSEXTENT:       500  CPU:      no limit
MAXJOBS:  0   MAXACCTJOBS:      0  PGFLQUOTA:  10000
Privileges:
  TMPMBX NETMBX
```

Figure 2-1: Sample Report for a Typical User Record With Many Default Values

Figure 2-2 illustrates a brief report that is produced for all the users with the same group number in their UIC.

```
UAF>SHOW [014,*] /BRIEF
       Owner             Username        UIC     Account   Privs Pri Default Directory
JOSEPH ROBIN             ROBIN        [014,006] VMS       Normal  4 SYS$USER:[ROBIN]
DAVID J. JONES           JONES        [014,011] VMS       Normal  4 SYS$USER:[JONES]
BOBBY WRENS              WRENS        [014,112] VMS       Normal  4 SYS$USER:[WRENS]
```

Figure 2-2: Sample Brief Report

Figure 2-3 illustrates the user authorization record for user WELCH. Observe that WELCH is a captive user who cannot change passwords, does not receive announcements of new mail or the welcome message when logging in. His login command file, SECUREIN.COM is presumably a captive command procedure, which controls all of his operations. (Such a command procedure never exits, but performs operations for its user and logs him out when appropriate.) The CAPTIVE flag prevents WELCH from escaping control of the command procedure by using CTRL/Y or other means. Furthermore, he is restricted to logging in between the hours of 3:00 A.M. and 11:59 P.M. on weekdays and 7:00 A.M. and 6:59 P.M. on weekends. Although he is allowed to use dial-up lines during the week, he is not allowed to log in over the network then. On weekends he is further restricted so that he cannot dial in or use the DCL command SET HOST.

```
UAF>SHOW WELCH
Username:  WELCH            Owner:  ROB WELCH
Account:   VMS              UIC:    [011,051]
CLI:       DCL              LGICMD: SECUREIN.COM
Default Device: SYS$USER:
Default Directory: [WELCH]
Login Flags: CAPTIVE DISNEWMAIL DISWELCOME
Primary days:   Mon Tue Wed Thu Fri          DISNETWORK
Secondary days:                     Sat Sun  DISDIALUP DISNETWORK
   00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
P- -- -- -- ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok
S- -- -- -- -- -- -- -- ok ok ok ok ok ok ok ok ok ok ok ok -- -- -- -- --
PRIO:      4  BYTLM:          4096  BIOLM:          6
PRCLM:     2  PBYTLM:            0  DIOLM:          6
ASTLM:    10  WSDEFAULT:       150  FILLM:         20
ENQLM:    10  WSQUOTA:         200  SHRFILLM:       0
TQELM:    10  WSEXTENT:        500  CPU:      no limit
MAXJOBS:   0  MAXACCTJOBS:       0  PGFLQUOTA:  10000
Privileges:
  TMPMBX NETMBX
```

Figure 2-3: Sample Report of a Restricted User Record

2.3 ERROR MESSAGES

The VAX/VMS System Messages and Recovery Procedures Manual lists the messages issued by AUTHORIZE, and provides explanations and suggested user actions.

CHAPTER 3

BACKUP UTILITY

The Backup Utility, or BACKUP, allows users to create back-up copies of files and directories from Files-11 structure level 1 and 2 volumes and to restore them. The utility can be used to back up entire volume sets in one operation or to perform selective back-ups by file or date. Wild cards are available, as well as several file selection qualifiers.

The Backup Utility can perform the following operations:

- Save disk files as data in a file created by BACKUP on disk or magnetic tape. Files created by the Backup Utility are called save sets. Only the Backup Utility can interpret the data in a save set.
- Restore disk files from a BACKUP save set.
- Copy disk files to disk.
- Compare disk files or files in a BACKUP save set with other disk files.
- List information about files in a BACKUP save set to an output device or file.

This chapter has four major sections.

- Section 3.1, The BACKUP Command, describes the command format and qualifiers used by the Backup Utility.
- Section 3.2, How to Use BACKUP, describes the forms of the DCL command BACKUP.
- Section 3.3, BACKUP Operations for System Managers and Operators, outlines BACKUP operations described in the _VAX/VMS System Management and Operations Guide_.
- Section 3.4, BACKUP Operation Examples, contains tables showing examples of commands used to perform the various BACKUP operations.

Appendix B contains information on the BACKUP media formats.

The Backup Utility is intended primarily for use by system managers and operators; however, it can be used by individual users to make personal back-up copies and to transport files between VAX/VMS systems. This chapter describes all qualifiers used in BACKUP operations and the elementary BACKUP procedures. BACKUP procedures for system managers and operators are outlined in Section 3.3 and described in the _VAX/VMS System Management and Operations Guide_.

These include descriptions of full volume back-up operations, sequential disk save sets, and the use of BACKUP journal files. Information on stand-alone BACKUP is found in the software installation guide for the VAX-11 processor on which you intend to use the utility.

Any user of BACKUP should be familiar with the VAX/VMS Command Language User's Guide.

3.1 THE BACKUP COMMAND

The format for the BACKUP command is:

```
     BACKUP input-specifier output-specifier
```

For most operations, the Backup Utility requires an input-specifier and an output-specifier. The specifiers can be either standard VAX/VMS file specifications or save-set-specifiers. A save-set-specifier is a file specification that identifies a file that contains data in BACKUP format (a save set). There is no default file type for save-set-specifiers.

You cannot use save-set-specifiers for both the input-specifier and output-specifier.

Save-set-specifiers can include node names; file specifications cannot.

Wild card characters may be used as part of file specifications (see Section 3.2.3).

BACKUP determines that it is to perform a save, restore, or copy operation by the devices named in the input-specifier and output-specifier. A save set may be located on any one of the following devices:

- Magnetic tape mounted with the DCL command MOUNT/FOREIGN
- A Files-11 disk (called a save set file)
- A Files-11 disk mounted with the DCL command MOUNT/FOREIGN (called a sequential disk save set)
- A file on a remote VAX/VMS node (called a network save set)

Normally, BACKUP assumes that a file on a Files-11 structure disk is a file specification; the /SAVE_SET qualifier is used to specify a save set on a Files-11 volume. A file on magnetic tape is always a save set. (Section 3.2.2 describes the media on which BACKUP records its save sets.) Table 3-1 shows the input-specifiers and output-specifiers used to perform save, restore, and copy operations.

BACKUP supports three types of qualifiers: BACKUP command qualifiers, which can be used anywhere in the command line; input-specifier qualifiers; and output-specifier qualifiers.

Table 3-1: Operations and Formats

Operation	Format
Save	BACKUP file-spec save-set-spec
Restore	BACKUP save-set-spec file-spec
Copy	BACKUP file-spec file-spec
Compare	BACKUP/COMPARE file-spec file-spec
	or
	BACKUP/COMPARE save-set-spec file-spec
List[1]	BACKUP/LIST save-set-spec

[1] The /LIST qualifier can also be used in conjunction with any other operation listed here.

3.1.1 BACKUP Command Qualifiers

The command qualifiers described below can be used anywhere in the BACKUP command line. Qualifiers followed by a dagger (†) can be used with stand-alone BACKUP; stand-alone BACKUP only performs image operations, unless you specify the /PHYSICAL qualifier. For information on image operations, see the description of the /IMAGE qualifier below.

Qualifiers	Defaults
/BRIEF†	/BRIEF
/COMPARE†	
/DELETE	
/FAST	
/FULL†	/BRIEF
/IGNORE=option	(see text)
/IMAGE†	
/INCREMENTAL	
/[NO]INITIALIZE†	(see text)
/JOURNAL[=file-spec]	
/LIST[=file-spec]†	(see text)
/[NO]LOG†	/NOLOG
/PHYSICAL†	
/RECORD†	
/[NO]TRUNCATE†	/NOTRUNCATE
/VERIFY†	
/VOLUME=n†	

/BRIEF

Indicates that brief file information is to be given, consisting of the file name, type, and version number; the size of the file in blocks; and the creation date of the file. This qualifier is valid only when used with the /LIST qualifier. The /BRIEF qualifier is the default format for output produced by the /LIST qualifier.

The /BRIEF qualifier is ignored unless the /LIST command qualifier is specified.

/COMPARE

Indicates that a compare operation is to be done. (No save, restore, or copy operation is performed.) The comparison may be between files on disks or between a save set and files on disk. If a comparison is to be made between files on disks, both input and output specifiers must refer to Files-11 media. If a comparison is to be made between a save set and files on disk, the input-specifier must be a save set and the output-specifier must be a Files-11 volume. Disk volumes must be mounted as Files-11 volumes. Tape volumes must be mounted using the /FOREIGN qualifier. Compare operations may also be performed using the /PHYSICAL qualifier to compare physical save sets against disk volumes, or to compare two disk volumes (a physical compare operation compares disk volumes in terms of physical blocks). In this case, disk volumes must be mounted using the /FOREIGN qualifier.

Note that the /COMPARE qualifier should be used only to compare a save set against original files or to compare files or volumes copied using BACKUP. BACKUP processes files by blocks. Comparing files that were not produced by BACKUP might cause mismatch errors.

/DELETE

Specifies that a BACKUP save operation is to delete the selected files after the save set has been successfully processed. Sufficient user privilege is required to delete files.

/FAST

Processes an input Files-11 medium using a fast file scan. The fast file scan reads the index file on the medium and creates a table of the files that match the qualifiers you have specified. You must have write access to the INDEXF.SYS file on the volume. The /FAST command qualifier is most effective when most files on a volume are selected by the input file specifier.

/FULL

Indicates that full file information is to be given in the form produced by the DCL command DIRECTORY/FULL. This qualifier is valid only when used with the /LIST qualifier. If the /FULL qualifier is not specified, only the name, size, and creation date of files are listed.

/IGNORE=option

Specifies that a BACKUP save or copy operation is to override restrictions placed on files. The options are:

INTERLOCK Processes files that otherwise could not be processed due to file access conflicts. This option can be used to save or copy files that are currently open for writing. Note that no synchronization is made with the process writing the file, so the file data that is copied might be inconsistent with the input file, depending on the circumstances (for example, if another user was editing the file, the contents might change). When a file open for writing is processed, BACKUP issues the message: %BACKUP-W-ACCONFLICT, 'filename' is open for write by another user. The use of this option requires the user privilege SYSPRV, a system UIC, or ownership of the volume.

NOBACKUP — Saves or copies the contents of files that are marked with the NOBACKUP flag by the /NOBACKUP qualifier to the DCL command SET FILE. If this qualifier is not used, files marked with the NOBACKUP flag are processed, saving all information necessary to recreate the file, except the actual contents of the file.

/IMAGE

Directs BACKUP to perform an image operation. An image operation processes the whole volume or volume set, and allows users to create a functionally equivalent copy of the input volume or volume set. To use the /IMAGE command qualifier, you need write access to both the volume's index file (INDEXF.SYS) and the bit map file (BITMAP.SYS). The file specifications used in an image operation must contain only device names.

In save and copy operations you cannot use file selection qualifiers with the /IMAGE command qualifier. All files on the input disk volumes are saved or copied; this includes reserved files and lost files (files that have no directory entry). Files that are entered in more than one directory are saved or copied only once. The directory specification stored in the save set for these files is that of the first directory in which the file was encountered. If you perform a file-by-file restore operation on these files, you must take this into consideration (see Section 3.3.2.1).

In restore and copy operations every file is restored or copied. Each output volume must be mounted using the /FOREIGN qualifier. The new volume is a functionally equivalent copy of the input volume; however, the file placement will change. In volume sets (more than one volume), the number of devices in the output specifier must be equal to the number of volumes in the input volume set. If you use the /IMAGE qualifier in a restore operation, the save set must have been created using the /IMAGE qualifier. You can also perform a selective restore (without the /IMAGE qualifier) from a save set that was created using the /IMAGE qualifier.

An image restore or copy operation initializes the output volume or volume set. The initialization data comes from the input volume unless the /NOINITIALIZE qualifier is used. Using the /NOINITIALIZE qualifier directs BACKUP to initialize the output volume using volume initialization data on that volume. You cannot change structure level in an image restore or copy.

/INCREMENTAL

Allows users to restore a disk volume from a save set created with the /IMAGE qualifier and a series of incremental save sets. When processed correctly, the output disk volume will contain the same files it contained when the most recent incremental back-up was performed. Files that were deleted in the period in which the series of incremental back-ups were performed are not created on the output disk volume. The /INCREMENTAL qualifier is valid only in restore operations. The output specifier must specify a device only; file specifications are not allowed.

For best results you should maintain full volume back-ups (using the /IMAGE qualifier) and perform frequent incremental back-ups. To restore the volume, restore the volume from the image save set using the /IMAGE and /RECORD qualifiers. The /RECORD qualifier directs BACKUP to write the date the volume was backed up into the back-up date field of the file header of each newly created file and directory. Then, apply the incremental save sets, using the /INCREMENTAL qualifier; start with the most recent save set. The order in which you apply the incremental save sets will not affect the final structure of the output volume; however, the most efficient order in which to apply the incremental save sets is reverse chronological order.

The /INCREMENTAL qualifier can be used only on Files-11 Structure Level 2 volumes.

/INITIALIZE
/[NO]INITIALIZE

Initializes an output disk volume. This qualifier is only valid when used with the /IMAGE qualifier on restore or copy operations, or when writing to a sequential disk save set. The default is /INITIALIZE.

When used with the /IMAGE qualifier, the /INITIALIZE qualifier directs BACKUP to initialize the volume using volume initialization data from the input volume. The /NOINITIALIZE qualifier directs BACKUP to reinitialize the output volume using the existing initialization data on that volume; the output volume must have been previously initialized as a Files-11 volume. Any existing data on the volume is lost. The structure level of the volume must be the same structure level as the save set being restored.

When used in writing to sequential disk save sets, the /INITIALIZE qualifier directs BACKUP to reinitialize the output volume. The /NOINITIALIZE qualifier directs BACKUP not to reinitialize the volume. Its use is analogous to the use of the BACKUP/NOREWIND command with magnetic tape.

/JOURNAL[=file-spec]

Specifies that a BACKUP save operation is to create, or append information to, a BACKUP journal file. The BACKUP journal file contains a record of back-up operations and the saved files.

If the specified journal file does not exist, it is created; if the journal file does exist, the new journal information is appended to the existing journal file. If you want to start a new journal file, you can also create a zero-length file (using the DCL command CREATE or a text editor). If the file specification is omitted, the default is SYS$DISK:BACKUP.BJL; if the file type is omitted, the default is BJL. The file specification cannot specify a node name.

The /JOURNAL qualifier can also be used with the /LIST qualifier to list the contents of a BACKUP journal file. You must not specify an input or output specifier with a BACKUP/JOURNAL/LIST command. The file selection qualifiers /BEFORE, /SINCE, /EXCLUDE, and /SELECT can be used with the /LIST qualifier to search for specific files. Note that the /BEFORE and /SINCE qualifiers refer to the time at which the save set was created, not the time at which the saved files were created.

/LIST[=file-spec]

Lists the names of files contained in a save set. You can use either the /BRIEF or /FULL qualifier with the /LIST qualifier. The /BRIEF qualifier directs BACKUP to list each file's size in blocks, and its creation date. The /FULL qualifier directs BACKUP to list additional information about each file in the form of the DCL command DIRECTORY/FULL. The default is /BRIEF. If no file specification is given, the default is SYS$OUTPUT. You can use the /LIST qualifier with any operation (save, restore, copy or compare) or by itself. If the /LIST qualifier is not being used in a save, restore, copy, or compare operation, the input specifier must refer to a save set, and the output specifier must be omitted. When you use the /LIST qualifier with stand-alone BACKUP, the file specification must reference either a terminal or a printer. Table 3-6 contains examples of the list operation.

/LOG
/[NO]LOG

Displays at SYS$OUTPUT the file specification of each file processed. The default is /NOLOG.

/PHYSICAL

Specifies that a BACKUP operation is to ignore any file structure on the volume. BACKUP saves, restores, copies, or compares the entire volume in terms of physical blocks.

An output disk must not have a bad block in any location that corresponds to a good block on the input disk. In addition, an output disk must be the same type of device as the input disk; for example, a BACKUP/PHYSICAL operation cannot be performed between an RP05 input disk and an RP06 output disk.

A save set written using the /PHYSICAL qualifier can only be read as a physical save set; conversely, a file-structured save set can only be read with file-structured restore or compare operations.

An output disk that is to be treated as a physical volume must have been mounted using the DCL command MOUNT/FOREIGN; an input disk must be mounted using the DCL command MOUNT/FOREIGN, or the user must have the user privilege LOG_IO. The file specification for a physical volume can contain only a device name.

/RECORD

Records the current date and time in the back-up date field of each file header once the save set is successfully processed. The /RECORD qualifier can be used only on Files-11 Structure Level 2 volumes. The user privilege SYSPRV is required to use the /RECORD qualifier on files other than those with your UIC.

If the /RECORD qualifier is used in a restore operation, the date and time that the save set was created is written in the back-up date field of the file header of each restored file.

/TRUNCATE
/[NO]TRUNCATE

Controls whether a copy or restore operation truncates a sequential output file at the end-of-file when creating it. By default, a copy or restore operation uses the allocation of the input file to determine the size of the output file.

/VERIFY

Performs data verification after backing up, restoring, or copying by comparing input and output media. The /VERIFY qualifier does not work on a restore or copy operation when the /NEW_VERSION output file qualifier is also used (see Section 3.1.3.1).

/VOLUME=n

Indicates that a specific disk volume in a disk volume set is to be processed. The /VOLUME qualifier is only valid when used with the /IMAGE qualifier.

Without the /VOLUME qualifier, a BACKUP copy operation involving a disk volume set of n volumes would require 2*n disk drives (since both volume sets would have to be fully mounted). However, a BACKUP operation using the /VOLUME qualifier requires only n+1 drives.

In save operations the save set contains the segments of the files located on the specified volume. The input volume set must be fully mounted. The save set can be restored only with the /VOLUME qualifier.

In copy operations the output volume is a functionally equivalent copy of the selected relative volume.

In restore operations the input save set must have been created using the /IMAGE qualifier. The input save set can be either an image save set of a full disk volume set, or a selected volume save set created with the /VOLUME qualifier. You cannot use the /NOINITIALIZE qualifier in a restore operation with the /VOLUME qualifier.

In a selected volume compare operation between two disk volume sets, both volume sets must be fully mounted. In a selected volume compare operation between a save set on tape and a disk volume set, the disk volume set must be fully mounted.

3.1.2 Input Qualifiers

There are two types of input qualifiers: file selection qualifiers and input save-set qualifiers. The file selection qualifiers can be used to select files from either a Files-11 volume or a BACKUP save set. Input save set qualifiers can only be used with an input save set.

Note that the input qualifiers are positional. They must appear after the input specifier; if they appear elsewhere in the BACKUP command, they might be misinterpreted.

3.1.2.1 Input File Selection Qualifiers

Qualifiers	Defaults
/BACKUP	
/BEFORE=time	
/CONFIRM	
/CREATED	
/EXCLUDE=(file-spec[,...])	
/EXPIRED	
/MODIFIED	
/OWNER_UIC[=[uic]]	(see text)
/SINCE=time	

/BACKUP

Selects files according to the back-up date written by BACKUP/RECORD (see Section 3.1.1). This qualifier is valid only when used with the /BEFORE or /SINCE qualifier. Only files that have been backed up before or since the date specified with the /BEFORE or /SINCE qualifier are selected. The /BACKUP qualifier should not be used with the /CREATED, /EXPIRED, or /MODIFIED qualifier. The /BACKUP qualifier can be used only with Files-11 Structure Level 2 volumes.

/BEFORE=time

Processes files dated earlier than the specified time. You can use the /BACKUP, /CREATED, /EXPIRED, or /MODIFIED qualifier with the /BEFORE qualifier; /MODIFIED is the default. The time option can be an absolute time with the format [dd-mmm-yyyy[:]][hh:mm:ss.c], or one of the following:

BACKUP	The back-up date of the file written by a previous BACKUP/RECORD operation (available only on Files-11 Structure Level 2 volumes)
TODAY	The current day, month, and year at 00:00:00.0 o'clock
TOMORROW	24 hours after 00:00:00.0 o'clock today
YESTERDAY	24 hours before 00:00:00.0 o'clock today

/CONFIRM

Prompts on SYS$COMMAND for confirmation before processing each file. If you want the file to be processed, type Y or YES, and press RETURN; typing anything else directs BACKUP not to process the file.

/CREATED

Selects files by their creation date when specified with either the /BEFORE or /SINCE qualifier. This qualifier should not be used with the /BACKUP, /EXPIRED, or /MODIFIED qualifiers.

/EXCLUDE=(file-spec[,...])

Omits a file, or files, from selection. You must not use device specifications in the file specification. You can use wild card characters in the file specification. Separate multiple file specifications with commas and enclose the list in parentheses.

Note that BACKUP does not apply temporary file specification defaults within the list. Each file specification independently takes its defaults from the file specification [000000...]*.*;*. Zero and negative version numbers (latest and relative version selection) cannot be used with the /EXCLUDE qualifier.

/EXPIRED

Selects files according to the expiration date when specified with either the /BEFORE or /SINCE qualifier. This qualifier should not be used with the /BACKUP, /CREATED, or /MODIFIED qualifiers.

/MODIFIED

Selects files according to the modification date when specified with either the /BEFORE or /SINCE qualifier. This qualifier should not be used with the /BACKUP, /CREATED, or /EXPIRED qualifiers.

/OWNER_UIC[=[uic]]

Selects files owned by the specified UIC. If the UIC is not specified with the /OWNER_UIC qualifier, the default is the UIC of the current process. If the /OWNER_UIC qualifier is omitted, all UICs on the volume are processed. You must specify the UIC in the following format:

[g,m]

g is an octal number in the range 0 through 377 representing the group number

m is an octal number in the range 0 through 377 representing the member number

The brackets ([ ]) are required in the UIC specification.

/SINCE=time

Processes files dated later than or at the specified time. You can use the /BACKUP, /CREATED, /EXPIRED, or /MODIFIED qualifiers with the /SINCE qualifier; /MODIFIED is the default. The option can be an absolute time with the format [dd-mmm-yyyy[:]][hh:mm:ss.c], or one of the following:

BACKUP	The back-up date of the file written by a previous BACKUP/RECORD operation (available only on Files-11 Structure Level 2 volumes)
TODAY	The current day, month, and year at 00:00:00.0 o'clock
TOMORROW	24 hours after 00:00:00.0 o'clock today
YESTERDAY	24 hours before 00:00:00.0 o'clock today

3.1.2.2 Input Save-Set Qualifiers - The command qualifiers listed below are used with input save sets. Qualifiers followed by a dagger (†) can be used with stand-alone BACKUP.

Qualifiers	**Defaults**
/[NO]CRC†	/CRC
/[NO]REWIND†	/REWIND
/SAVE_SET†	
/SELECT=(file-spec[,...])	

/CRC
/[NO]CRC

Specifies whether the software Cyclic Redundancy Check (CRC) is to be made. The default is /CRC. To disable checking, specify /NOCRC; note, however, that use of the /NOCRC qualifier reduces processing time at the risk of increasing data error.

/REWIND
/[NO]REWIND

Specifies whether an input tape is to be rewound before processing. /REWIND is the default.

/SAVE_SET

Directs BACKUP to treat the input file as a BACKUP save set. Normally, an input specifier that refers to disk is treated as a Files-11 medium, and an input specifier that refers to tape is treated as a BACKUP save set. The /SAVE_SET qualifier allows you to refer to a BACKUP save set on a Files-11 medium located on either a local system or on a remote VAX/VMS system via DECnet.

The /SAVE_SET qualifier is also used with sequential disk save sets. For more information on sequential disk save sets, see Section 3.2.2.4 and the VAX/VMS System Management and Operations Guide.

/SELECT=(file-spec[,...])

Selects a specific file, or files, for processing. You cannot use device specifications in the file specification, but you can use wild card characters in the file specification. Separate multiple file specifications with commas and enclose the list in parentheses.

Note that BACKUP does not apply temporary file specification defaults within the list. Each file specification independently takes its defaults from the file specification [000000...]*.*;*. Zero and negative version numbers (latest and relative version selection) cannot be used with the /SELECT qualifier.

3.1.3 Output Qualifiers

Like the input qualifiers, there are two types of output qualifiers: output file qualifiers and output save set qualifiers. The output file qualifiers apply to files on a Files-11 volume. Output save set qualifiers can only be used with an output save set.

Note that the output qualifiers are positional. They must appear after the output specifier; if they appear elsewhere in the BACKUP command, they might be misinterpreted.

3.1.3.1 Output File Qualifiers - The command qualifiers listed below are used with output files.

Qualifiers	Defaults
/NEW_VERSION	
/OVERLAY	
/OWNER_UIC[=option]	/OWNER_UIC=DEFAULT
/REPLACE	

/NEW_VERSION

If BACKUP attempts to copy or restore a file when a file with an identical directory name, file name, type, and equal or higher version number already exists, a new file is created with the same name and type, and a version number one higher than the highest existing version. If you omit this qualifier and the version number is identical, BACKUP reports an error in copying the file and no replacement is made.

Note that when copying or restoring files using the /NEW_VERSION qualifier, files are processed in decreasing version number order, and created in ascending version number order. This causes the version numbers of files to be inverted.

Because this qualifier causes file version numbers to change, its use with the /COMPARE or /VERIFY qualifiers will cause unpredictable results.

/OVERLAY

If BACKUP attempts to copy or restore a file when a file with an identical directory name, file name, type, and version number already exists, the new version of the file is written over the existing version. The physical location of the file on disk does not change. If the /OVERLAY qualifier is specified, and the new file is larger than the one already present, BACKUP allocates more free blocks on the disk, and extends the file; if the new file is smaller than the one already present, BACKUP reduces the number of blocks used. If you omit this qualifier, BACKUP reports an error in copying the file and no replacement is made.

/OWNER_UIC[=option]

Specifies the owner UIC of the created files by one of the following options:

DEFAULT — Sets the UIC to the user's current default UIC. This is the default option if /OWNER_UIC[=option] is not specified.

ORIGINAL — Sets the UIC to the UIC of the original copy of the file. If you specify /OWNER_UIC without specifying an option, /OWNER_UIC=ORIGINAL is the option taken. To use this option, you need the SYSPRV privilege, you must own the volume, or the UIC must be your own.

PARENT — Sets the UIC to the owner UIC of the directory in which the files are being created. To use this option, you need the SYSPRV privilege, you must own the volume, or the parent UIC must be your own.

[uic] — Sets the UIC to the specified UIC. To use this option, you need the SYSPRV privilege, you must own the volume, or the UIC must be your own.

You must specify the UIC in the following format:

[g,m]

g is an octal number in the range 0 through 377 representing the group number

m is an octal number in the range 0 through 377 representing the member number

The brackets ([ ]) are required in the UIC specification.

/REPLACE

If BACKUP attempts to copy or restore a file when a file with an identical directory name, file name, type, and version number already exists, the existing version of the file is deleted and a new version is created. If you omit this qualifier, BACKUP reports the error and no replacement is made.

3.1.3.2 Output Save-Set Qualifiers - The command qualifiers listed below are used with output save sets. All of these qualifiers can be used with stand-alone BACKUP, as indicated by the dagger (†) following each.

Qualifiers	Defaults
/BLOCK_SIZE=n†	(see text)
/COMMENT=string†	
/[NO]CRC†	/CRC
/DENSITY=n†	(see text)
/GROUP_SIZE=n†	/GROUP_SIZE=10
/LABEL=(string[,...])†	
/OWNER_UIC=[uic]†	(see text)
/PROTECTION[=(code)]†	(see text)
/[NO]REWIND†	/REWIND
/SAVE_SET†	

/BLOCK_SIZE=n

Specifies the output block size in bytes. The minimum is 2048 bytes; the maximum is 65535 bytes. The actual block size written is adjusted using the constraints of the BACKUP format. On tapes, the Backup Utility ignores any block size defined by the /BLOCKSIZE qualifier to the DCL command MOUNT. A block cannot be rounded up over the maximum block size. The default block size for tape is 8464 bytes; the default for disk is 32528 bytes.

If the block size is set to a large value for a save set on tape, it is possible for the tape to run off its reel or for a large number of write errors to be logged. If this occurs, avoid using large block sizes. If the problem recurs with the same tape, avoid using the tape for future BACKUP operations.

/COMMENT=string

Places a comment in an output save set. If the string is longer than one word, or if it contains nonalphanumeric characters, it must be surrounded by quotation marks (").

/CRC
/[NO]CRC

Specifies whether the software CRC is to be computed and stored in the data blocks of the save set. The default is /CRC. To disable checking, use /NOCRC; note, however, that use of the /NOCRC qualifier reduces processing time at the risk of increasing data error.

/DENSITY=n

Specifies the recording density of the output tape. The value given must be one that your tape hardware supports. If you omit this qualifier, the default density is the current one on the drive addressed.

/GROUP_SIZE=n

Specifies the number of blocks to be used in each XOR redundancy group. The value must be in the range 0 through 100. If the value is 0, no redundancy blocks are created. One read error in each redundancy group can be corrected using the redundancy information. If you omit this qualifier, the default value is 10. For more information on redundancy groups, see Appendix B.

/LABEL=(string[,...])

Specifies the volume label for a save set written on tape or sequential disk. For save sets written on tape, the string must be from 1 through 6 alphanumeric characters.

For save sets written on sequential disk volumes, the string must be from 1 through 12 alphanumeric characters.

If you do not specify the /LABEL qualifier, the label will be derived from the save-set-name. In a multivolume disk save set, the volume set name will be the save-set-name. The label of each volume in the volume set will either be the label string or the label string derived from the save-set-name and followed by a two digit volume number, starting with 01.

If you specify a list of labels, BACKUP will label save set volume n with label n in the list of labels. If the list of labels is shorter than the number of volumes in the save set, BACKUP will generate labels for the remaining volumes using the first label in the list followed by a two-digit relative volume number. Note that for tape save sets, the volumes are counted, starting (at 01) with the tape on which the current save set started. Tape volume numbers are not maintained across multiple save sets written on multiple tapes.

/OWNER_UIC=[uic]

Specifies the owner UIC of the save set. If the /OWNER_UIC qualifier is omitted, the UIC of the current process is used. To use this qualifier on Files-11 save sets, you need the user privilege SYSPRV, or the UIC must be your own. The format of the UIC is shown in the description of the /OWNER_UIC input file selection qualifier (see Section 3.1.2.1).

/PROTECTION[=(code)]

Defines the protection to be applied to the save set. The code indicates the type of access (read, write, execute, and delete) available to the four categories of users (system, owner, group, and world). For more information on specifying the protection code, see the VAX/VMS Command Language User's Guide.

If the save set is written to either a Files-11 disk or a sequential disk, and the /PROTECTION qualifier is not specified, the default is the process default protection.

If the save set is written to tape and the /PROTECTION qualifier is not specified, no protection is given to the tape. If the /PROTECTION qualifier is specified without specifying a code, the default is the process default protection.

Any protection categories not specified are defaulted to your process default protection.

/REWIND
/[NO]REWIND

Specifies whether an output tape is to be rewound before processing begins. /REWIND is the default. If you plan to put a save set on a tape that already contains a save set, you must use the /NOREWIND qualifier. Otherwise, the new save set will be written over the previous contents of the tape. The /NOREWIND qualifier directs BACKUP to append the new save set following the last existing file on the tape, rather than write the new save set at the current position.

/SAVE_SET

Directs BACKUP to treat the output file as a BACKUP save set. Normally, an output specifier that refers to disk is treated as a Files-11 medium, and an output specifier that refers to tape is treated as a BACKUP save set. The /SAVE_SET qualifier allows you to create a BACKUP save set on a Files-11 medium located on either a local system or on a remote VAX/VMS system with DECnet.

The /SAVE_SET qualifier is also used with sequential disk save sets. For more information on sequential disk save sets see Section 3.2.2.4 and the VAX/VMS System Management and Operations Guide.

3.2 HOW TO USE BACKUP

This section describes BACKUP media; how you use output wild card characters; and how to save, restore, copy, and compare files. Tables 3-2 through 3-6, in Section 3.4, summarize BACKUP operations and contain examples of commands used to accomplish the operations.

3.2.1 Protecting a BACKUP Save Set

The output save set qualifiers /PROTECTION and /OWNER_UIC allow you to specify the protection and owner UIC for a save set. These qualifiers can be used with save sets stored on magnetic tape, sequential disk, or Files-11 disk. These qualifiers will prevent nonprivileged users from mounting a save set volume or accessing a Files-11 save set. Note, however, that once a user has access to a save set, a user can read any data contained in the save set.

3.2.2 BACKUP Media

You can write save sets to the following types of media:

- Magnetic tape
- Files-11 disk
- Remote Files-11 disk
- Sequential disk

3.2.2.1 Magnetic Tape Save Sets - Magnetic tape must be mounted with the DCL command MOUNT/FOREIGN. (Information on the MOUNT command can be found in the VAX/VMS Command Language User's Guide). Magnetic tape volumes do not need to be initialized. You can use more than one tape device at a time to save or restore data. This allows processing to continue on another tape while the one most recently used is rewinding. If no save set name is given for an input magnetic tape, the first save set encountered on the tape will be read. For more information concerning magnetic tape volumes, see Section 3.2.5.

3.2.2.2 Files-11 Disk Save Sets - To write save sets on Files-11 disk, you must use the /SAVE_SET output save set qualifier. This indicates to BACKUP that you want to create a save set on the output volume, rather than a copy of the selected files. The disk must be mounted as a Files-11 volume; all volumes in a volume set must be mounted.

3.2.2.3 Remote Save Sets - You can create or access a save set file on a remote node by specifying a node name (and an access control string, if one is required) in a save set specifier.

3.2.2.4 Sequential Disk Save Sets - Sequential disk save sets allow you to treat a Files-11 disk volume in a manner similar to the way you treat a magnetic tape save set. The primary advantage of using sequential disk save sets is that you can mount multivolume save sets one volume at a time. This is particularly useful on systems without tape that have a large fixed-media disk and a small removable disk. When one sequential disk is full, BACKUP prompts you for another disk, as it would for save sets on magnetic tape.

To write sequential disk save sets, you must mount the output disk using the DCL command MOUNT/FOREIGN. Although the disk is mounted with the /FOREIGN qualifier, the Backup Utility manages the disk using Files-11 structure. When you perform a save operation, you must use the /SAVE_SET output save set qualifier. You must not specify a directory name for the save set; save sets are entered in the disk's Master File Directory. The user privilege LOG_IO is required to read or write a multiple volume sequential disk save set. You can use more than one disk device at a time to save or restore data. This allows processing to continue on another disk while the one most recently used is spinning down. The Backup Utility will initialize sequential disk volumes before writing to them. To prevent the disk from being initialized, and to save any existing data on the output disk volume, use the /NOINITIALIZE qualifier. If you use the /NOINITIALIZE qualifier, the following restrictions apply to the output volume:

- The disk must be Files-11 Structure Level 2
- The disk must not be part of a volume set
- The cluster factor of the disk must be 1
- The free space on the disk cannot be fragmented into more than 100 pieces
- The index file cannot be extended
- The Master File Directory cannot be extended

The effect of these restrictions is to limit the number of save sets on a volume to approximately 20.

In general, the /NOINITIALIZE qualifier is most useful in adding a save set to an existing sequential disk volume.

You can read a sequential disk save set in either of two ways: as a sequential disk save set (mounted with the /FOREIGN qualifier) or as a normal Files-11 save set.

If a save set is read as a sequential disk save set, one volume can be mounted at a time (as you can in creating a sequential disk save set). The default directory for the save set file specification is the directory [000000] (the disk's master file directory).

If a save set is read as a normal Files-11 save set, all volumes of the save set must be mounted (as is true for any Files-11 volume set). The default directory is your process default directory, so to read the save set, you must specify the directory [000000].

Save set files written to disk using the file system can also be read in sequential mode. You must specify the directory from which the save set is to be read. This mode of operation is useful when using Stand-alone BACKUP. If the save set was written to a volume set using Files-11 disk structure, you must observe the following rules:

- All volumes of the volume set must be mounted.
- You must mount the volumes with the /FOREIGN qualifier. (If you are using stand-alone BACKUP, the volumes are implicitly mounted.)
- When you specify the volumes in the input specifier, you must specify the names of the devices in the same order as the relative volume number of the volumes mounted on those devices. For example, the volumes named USER01 and USER02 in a volume set are mounted on two drives, DRA3 and DRA5. The save set specifier for the volume set must be DRA3:[directory]save-set-name.ext;v, DRA5:.

For more information on sequential disk save sets, see the _VAX/VMS System Management and Operations Guide_.

BACKUP command and file qualifiers are described in Section 3.1.

3.2.3 Wild Card Characters

The Backup Utility allows a number of wild card characters to be used as part of file specifications. (For introductory information on wild card characters, see the _VAX/VMS Command Language User's Guide_.) Directories, file names, types, and version numbers can all be represented by wild card characters. Note that omitting the version number causes all versions of the specified file to be processed. Omitted file names, types, or version numbers are assumed to be wild card characters. Wild card characters cannot be used to specify save sets. When used in input-specifiers, or with the /SELECT and /EXCLUDE qualifiers, any valid DCL wild card can be used in directories. Note, however, that the latest version (;0) and relative versions (;-n) are treated as the all-version wild card specification (;*) with the /EXCLUDE and /SELECT qualifiers.

3.2.4 Examples of BACKUP Operations

The following sections contain examples that illustrate typical commands used to save, restore, copy, and compare files. Assume that a tape has been mounted using the /FOREIGN qualifier, and has the logical name TAPE:.

3.2.4.1 Saving Files - To create a back-up file of the current default directory, type:

```
$ BACKUP *   TAPE:NOV12SAVE.BCK
```

All the files in the current default directory will be saved in a save set named NOV12SAVE.BCK (a reminder that the save operation was done on November 12). Note that the save-set-name can be any valid VAX/VMS file name and type.

To create a back-up file that contains all files and subdirectories of a particular directory, type:

```
     $ BACKUP [LYKINS...]*    TAPE:NOV13SAVE.BCK
```

All files in the directory [LYKINS], all subdirectories of [LYKINS], and all the files in those subdirectories will be included in the save set NOV13SAVE.BCK.

To create a BACKUP file of a single file, type:

```
     $ BACKUP STRATCOL1.DAT   TAPE:STRATDAT1.BCK
```

The file STRATCOL1.DAT will be saved in the save set STRATDAT1.BCK.

If you specify a save set on more than one tape or sequential disk device, you must specify the save-set-name only with the first device; separate the other devices with commas. For example:

```
     $ BACKUP * MTA0:14AUG,MTA1:
```

3.2.4.2 Restoring Files

- To restore all files in the save set to the current default directory, type:

```
     $ BACKUP TAPE:NOV12SAVE.BCK []
```

To restore files in subdirectories, type:

```
     $ BACKUP TAPE:NOV12SAVE.BCK [LYKINS...]
```

To restore a specific file from a save set, you can use the /SELECT qualifier. In the following example, the file STRAT1.DAT in the directory [LYKINS.GLENDO] has been accidentally deleted. The user has previously saved the file in the save set NOV2SAVE.BCK.

```
     $ MOUNT/FOREIGN MTA0:
     %MOUNT-I-MOUNTED, NOV25A mounted on _MTA0:
     $ BACKUP
     $_From:        MTA0:NOV2SAVE.BCK/SELECT=[LYKINS.GLENDO]STRAT1.DAT;5
     $_To:          STRAT1.DAT;5
     $_DIRECTORY STRAT1.DAT
     Directory [LYKINS.GLENDO]

     STRAT1.DAT;5

     Total of 1 file.
     $
```

If your save set encompasses more than one tape or sequential disk volume, it is possible to begin restore and compare operations without mounting the first volume of the save set. However, if you use the /IMAGE qualifier, processing must begin with the first volume. If the tape that you mount is not the first volume in the save set, you will receive the warning message:

```
     %BACKUP-W-NOT1STVOL, tape 'name' is not the start of a save set
```

When restoring a small number of files from a large save set, it is a good idea to use the /LOG qualifier. This enables you to monitor the files as they are restored. BACKUP will continue processing the save set until it reaches the end of the save set. Once the files you need have been restored, you can press CTRL/Y to terminate processing.

3.2.4.3 Copying Files

- The BACKUP copy operation can copy files, directories, directory structures, and disks. For example:

```
     $ BACKUP [LYKINS]UPLIFT.PAS;  [OWLCR]UPLIFT.PAS;1
```

This command directs BACKUP to copy the highest numbered version of file UPLIFT.PAS in the directory [LYKINS] to the directory [OWLCR] and to name the file UPLIFT.PAS;1.

3.2.4.4 Comparing Files

- You can use the BACKUP compare operation either to compare a save set with files on disk, or to compare files on disk with other files on disk. Note that the /COMPARE qualifier should be used only to compare a save set against original files or to compare files or volumes copied using BACKUP. BACKUP processes files by blocks. Comparing files that were not produced by BACKUP might cause mismatch errors.

To compare a save set with files on disk, type:

```
     $ BACKUP/COMPARE TAPE:2MAR1555.BCK [LYKINS]
```

This command directs BACKUP to compare the contents of the save set 2MAR1555.BCK with the directory [LYKINS]. The /COMPARE qualifier is described in Section 3.1.1.

To compare files on disk, type:

```
     $ BACKUP/COMPARE UPLIFT.EXE;3 UPLIFT.EXE;4
     %BACKUP-E-VERIFYERR, verification error for block 16 of WRKD$:[LYKINS]
     UPLIFT.EXE;4
```

In this example there is an inconsistency in block 16 between UPLIFT.EXE;4 and UPLIFT.EXE;3.

3.2.5 Multivolume Save Sets

When you create a save set on magnetic tape or sequential disk, you may be requested to mount another volume. When the end-of-tape is reached, or the disk is full, BACKUP puts the drive offline. (Note that not all types of disk drives can be put offline under program control.) If you are using magnetic tape, BACKUP rewinds the tape. You then receive the messages:

```
     %BACKUP-I-RESUME, resuming operation on volume 'number'
     %BACKUP-I-READYWRITE, mount volume 'number' on 'device' for writing
     Press return when ready:
```

Remove the full volume, and load another unused volume on the drive. Press the RETURN key when the volume has been loaded. On disk devices, press the RETURN key when the device is up to speed and the ready light is on.

When you are reading a save set, you receive messages in the form:

```
     %BACKUP-I-RESUME, resuming operation on volume 'number'
     %BACKUP-I-READYREAD, mount volume 'number' on 'device' for reading
     Press return when ready:
```

Remove the current volume from the drive, and load the next volume in the set. Press the RETURN key when the volume has been loaded (and the ready light is on).

After you mount a volume and press the RETURN key, processing continues normally.

You must have the user privilege LOG_IO to process multiple sequential disk volumes in this manner. No special privilege is required for tape.

If you are using magnetic tape or sequential disk volumes, you can specify more than one device in the command, and have a volume ready in the next device on the list. When using magnetic tape, you will get only one message:

```
    %BACKUP-I-RESUME, resuming operation on volume 'number'
```

You are always prompted for sequential disk volumes, because many disk drives cannot be unloaded under program control.

If you invoke the Backup Utility from a batch job, the tape mounting messages, READYREAD, READYWRITE, and WRITENABLE, are displayed at terminals enabled to receive operator's messages. Terminals enabled with REPLY/ENABLE=TAPES receive messages for tape operations, and terminals enabled with REPLY/ENABLE=DISKS receive messages for sequential disk operations. Terminals enabled with REPLY/ENABLE=CENTRAL receive messages for both operations. (For information on enabling terminals with the REPLY/ENABLE command, see the VAX/VMS Command Language User's Guide.)

The Backup Utility will continue processing when you reply to the requests using the DCL command REPLY/TO. If you respond with REPLY/PENDING, the Backup Utility will wait until a REPLY/TO command is entered. If you respond with REPLY/ABORT, the fatal error %BACKUP-F-OPERFAIL causes the utility to terminate. (For information on the DCL command REPLY see the VAX/VMS Command Language User's Guide.)

NOTE

> BACKUP requires the user privilege TMPMBX to interact with the operator from batch mode.

3.3 BACKUP OPERATIONS FOR SYSTEM MANAGERS AND OPERATORS

A number of BACKUP operations and qualifiers are particularly useful to system managers and operators. A short description of the operations and qualifiers is presented here; for a more complete description, see the VAX/VMS System Management and Operations Guide.

3.3.1 Incremental Back-ups

The /RECORD command qualifier directs the Backup Utility to record the current date and time as each file is saved or copied. When the /RECORD command qualifier is used in conjunction with the /SINCE=BACKUP input file selection qualifier in successive BACKUP operations, only files that were created or modified since the last back-up operation are processed.

3.3.2 Full-volume Back-ups

The /IMAGE qualifier directs BACKUP to save information necessary to reinitialize the disk volume and to save all files on the volume. When the /IMAGE qualifier is used in a restore or copy operation, the output volume is reinitialized; the restored volume is a functionally equivalent copy of the original volume. If the output volume has already been initialized as a Files-11 volume, the /NOINITIALIZE qualifier can be used to reinitialize the volume using the volume initialization parameters found on the volume.

3.3.2.1 Files With Multiple Directory Entries - When BACKUP handles files with multiple directory entries, its behavior varies, depending on the qualifiers used.

If the /IMAGE qualifier was used in both the save and restore operations (or on a copy operation), a file with multiple directory entries will retain its configuration. That is, multiple directory entries on an output volume will point to the same file.

If a volume is saved and restored without specifying the /IMAGE qualifier on either operation, a file with multiple directory entries will be replicated on the output volume. For each directory entry that existed on the input volume, a separate copy of the file will be created on the output volume.

If a volume is saved using the /IMAGE qualifier and is restored without the /IMAGE qualifier, a file with multiple directory entries will be restored under the first directory entry encountered; the remaining directory entries are not restored.

3.3.3 Restoring Entire Volumes and Applying Incremental Save Sets

To restore an entire disk volume, you must first restore the volume from the most recent full volume save set. To get correct results, you must restore the volume from an image save set (using the /IMAGE qualifier), and you must use the /RECORD qualifier. The /RECORD qualifier writes the date that the file was saved in the backup date field of each file header. The back-up date is used when applying incremental save sets.

For example, the following command would be used to restore a disk volume from an image save set:

```
     $ BACKUP/IMAGE/RECORD MTA0:11JNUSER.FUL DRA4:
```

Once the disk has been restored, the incremental save sets must be applied. To apply an incremental save set, you restore each save set using the /INCREMENTAL qualifier. The output specifier must specify a device only. For optimal processing it is best to restore the incremental save sets in reverse chronological order (start with the most recent incremental save set and work back to the first incremental save set created after the full save). For example, the following series of commands would be used to apply incremental save sets to the volume restored in the example above. The full volume save was made on June 11, 1982; the incremental saves were performed in the week following.

```
     $ BACKUP/INCREMENTAL MTA0:18JNUSER.INC  DRA4:
     $ BACKUP/INCREMENTAL MTA0:17JNUSER.INC  DRA4:
     $ BACKUP/INCREMENTAL MTA0:16JNUSER.INC  DRA4:
     $ BACKUP/INCREMENTAL MTA0:15JNUSER.INC  DRA4:
     $ BACKUP/INCREMENTAL MTA0:14JNUSER.INC  DRA4:
```

The volume should now contain the same files that it contained when the incremental save was performed on June 18.

The use of the /INCREMENTAL qualifier causes BACKUP to restore directory files. If the save operation that created the incremental save set used the /EXCLUDE file selection qualifier, directory entries will be created for nonexistent files. These must be cleaned up later with the Verify Utility (DCL command ANALYZE/DISK_STRUCTURE).

3.3.4 Stand-alone BACKUP

Stand-alone BACKUP is a stand-alone version of the Backup Utility. It uses a subset of the BACKUP qualifiers and can be bootstrapped from console block-storage device or from a Files-11 disk. (See Section 3.1 for qualifiers that can be used with stand-alone BACKUP.) To bootstrap stand-alone BACKUP from console block-storage device, see the software installation guide for your VAX-11 processor.

Before you bootstrap stand-alone BACKUP from a Files-11 disk, you must first copy files necessary to stand-alone BACKUP to the Files-11 disk. To copy the files, use the command procedure SYS$UPDATE:STABACKIT.COM. The command procedure will prompt you for a target device. The files are placed in the directory [SYSEXE]. If the directory does not already exist, it is created. You can have the target device allocated, and it can optionally be mounted. The user privilege SETPRV (or the user privileges CMKRNL, CMEXEC, LOG_IO, SYSNAM, VOLPRO, and OPER) is required to use STABACKIT.COM. Once It is running, the command procedure will list at your terminal the files as they are copied. When it is finished, the command procedure will type the message:

```
     Kit is complete.
```

To bootstrap stand-alone BACKUP from a Files-11 disk, use the nonstop bootstrap command for your VAX-11 processor. The bootstrap device must be the device on which the disk containing stand-alone BACKUP is mounted. For more information, see the installation guide for your VAX-11 processor.

The previous three sections discussed operations that are particularly useful for system managers and operators. The following two sections discuss operations that can be performed by any user.

3.4 BACKUP OPERATION EXAMPLES

Tables 3-2 through 3-6 are intended for users who need a quick reference to the BACKUP command and qualifiers. The action of the command is listed on the left. On the right is the format that the command should take and an example of the command.

Not all possible variations have been listed, but it should be possible to use the information given here to create the desired command.

Table 3-2 shows BACKUP command formats for save operations and some of the qualifiers associated with a save operation.

Table 3-2: Save Operation Quick Reference Table

Command Action	Command Format Command Example
Save a file to a save set on tape	$ BACKUP file-spec save-set-specifier $ BACKUP STRATDAT1.DAT MTA0:STRATDAT1.BCK
Save the most recent versions of files in a directory to tape	$ BACKUP [directory]*.*; save-set-specifier $ BACKUP [LYKINS...]*.*; MTA0:1409MAR17.BCK
Save a disk volume to a save set on tape	$ BACKUP/IMAGE ddcu: save-set-specifier $ BACKUP/IMAGE DBA1: MTA0:930FEB4.BCK
Save a disk volume to a multivolume save set on more than one tape drive	$ BACKUP/IMAGE ddcu: save-set-specifier,ddcu:... $ BACKUP/IMAGE DBA1: MTA0:17MAR.BCK,MTA1:
Save a list of files to a save set on tape	$ BACKUP file-spec,file-spec,... save-set-specifier $ BACKUP DBA1:[LYKINS...]*.PAS,DMA0:[DAKOTA...]*.PAS MTA0:PAS17MAR.BCK
Save a disk volume for incremental back-ups for the first time	$ BACKUP/RECORD/IMAGE/LOG ddcu: save-set-specifier $ BACKUP/RECORD/IMAGE/LOG DBA1: MTA0:925FEB4.BCK
Save a disk volume for incremental back-ups (not the first time)	$ BACKUP/RECORD/FAST/LOG ddcu:[*...]/SINCE=BACKUP save-set-specifier $ BACKUP/RECORD/FAST/LOG DBA1:[*...]/SINCE=BACKUP MTA0:928FEB4.BCK
Save an unstructured disk volume	$ BACKUP/PHYSICAL ddcu: save-set-specifier $ BACKUP/PHYSICAL DMA1: MTA0:935FEB4.BCK
Save a directory to a save set on a Files-11 disk	$ BACKUP [directory] save-set-specifier/SAVE_SET $ BACKUP [LYKINS] DBA2:[BACKUP]1609FEB3.BCK/SAVE_SET
Save a directory tree to a save set on tape	$ BACKUP [directory...] save-set-specifier $ BACKUP [LYKINS...] MTA0:1612FEB3.BCK
Save a directory tree to a save set on tape and output a listing	$ BACKUP/LIST=file-spec [directory...] save-set-specifier $ BACKUP/LIST=8SEP.LOG [LYKINS...] MTA0:8SEP.BCK

Table 3-3 shows BACKUP command formats for restore operations and some of the qualifiers associated with restore operations. In the examples in this table, it is assumed that save sets already exist on the tape and disk.

Table 3-3: Restore Operation Quick Reference Table

Command Action	Command Format/Command Example
Restore from save set on disk to a Files-11 disk with original UICs	$ BACKUP save-set-specifier/SAVE_SET ddcu:[*...]/OWNER_UIC=ORIGINAL
	$ BACKUP DBA2:[BACKUP]1025FEB2.BCK/SAVE_SET DBA1:[*...]/OWNER_UIC=ORIGINAL
Restore from a save set on tape to a Files-11 disk with original UICs	$ BACKUP save-set-specifier ddcu:[*...]/OWNER_UIC=ORIGINAL
	$ BACKUP MTA0:1618FEB2.BCK DBA1:[*...]/OWNER_UIC=ORIGINAL
Restore a selected file in a save set on tape to a Files-11 disk	$ BACKUP save-set-specifier/SELECT=file-spec file-spec
	$ BACKUP MTA0:1618FEB2.BCK/SELECT=[POUDRE]UPLIFT.PAS [GEO.PAS]UPLIFT.PAS
Restore files with a specific UIC to a Files-11 disk	$ BACKUP save-set-specifier/OWNER_UIC=[uic] file-spec
	$ BACKUP MTA0:1641FEB2.BCK/OWNER_UIC=[360,052] [LYKINS...]
Restore files to a Files-11 disk with a new UIC	$ BACKUP save-set-specifier file-spec/OWNER_UIC=[uic]
	$ BACKUP MTA0:1641FEB2.BCK [TESTS...]/OWNER_UIC=[100,150]
Restore files to a Files-11 disk; if file exists, create new version	$ BACKUP save-set-specifier file-spec/NEW_VERSION
	$ BACKUP MTA0:1641FEB2.BCK [LYKINS...]/NEW_VERSION
Restore files to a Files-11 disk; if file exists, replace with new version	$ BACKUP save-set-specifier file-spec/REPLACE
	$ BACKUP MTA0:1641FEB2.BCK [LYKINS...]/REPLACE
Restore files to a Files-11 disk excluding certain files	$ BACKUP save-set-specifier/EXCLUDE=file-spec file-spec
	$ BACKUP MTA0:1641FEB2.BCK/EXCLUDE=[LYKINS.PAS] [LYKINS...]
Restore a directory tree, placing files in a different sub tree	$ BACKUP save-set-specifier/SELECT=[directory...] [directory2...]
	$ BACKUP MTA0:1641FEB2.BCK/SELECT=[FIELD...] [LYKINS.NEWDATA...]
Restore a Files-11 volume from a physical save set	$ BACKUP/PHYSICAL save-set-specifier ddcu:
	$ BACKUP/PHYSICAL MTA0:26MAR.BCK DMA3:
Restore a Files-11 volume from an image save set	$ BACKUP/IMAGE save-set-specifier ddcu:
	$ BACKUP/IMAGE MTA0:17AUG.BCK DRA3:
Restore a Files-11 volume, changing its initialization parameters	$ INITIALIZE ddcu: volume-name/new-parameters $ MOUNT/FOREIGN ddcu: $ BACKUP/IMAGE save-set-specifier ddcu:/NOINITIALIZE/TRUNCATE
	$ INITIALIZE DBA1: UTTLPACK/CLUSTER=5 $ MOUNT/FOREIGN DBA1: $ BACKUP/IMAGE MTA0:17AUG.BCK DBA1:/NOINITIALIZE/TRUNCATE

Table 3-4 shows BACKUP command formats for copy operations, and some of the qualifiers associated with a copy operation.

Table 3-4: Copy Operation Quick Reference Table

Command Action	Command Format Command Example
Copy a directory tree to another directory tree	$ BACKUP [directory...] [directory...]
	$ BACKUP [DAKOTA...] [SUNDANCE...]
Copy a file to another file	$ BACKUP file-spec file-spec
	$ BACKUP LOGIN.COM [.SAVE]OLDLOGIN.COM
Copy a disk volume to another disk volume	$ BACKUP/IMAGE ddcu: ddcu:
	$ BACKUP/IMAGE DBA1: DBA2:
Copy a disk volume to another disk volume using the /PHYSICAL qualifier	$ BACKUP/PHYSICAL ddcu: ddcu:
	$ BACKUP/PHYSICAL DYA1: DYA2:
Copy two disk volume set using the /IMAGE qualifier	$ BACKUP/IMAGE volume-set-name ddcu:,ddcu:
	$ BACKUP/IMAGE USER$: DBA1:,DBA2:

Table 3-5 shows BACKUP command formats for compare operations, and some of the qualifiers associated with a compare operation.

Table 3-5: Compare Operation Quick Reference Table

Command Action	Command Format Command Example
Compare two Files-11 files	$ BACKUP/COMPARE file-spec file-spec
	$ BACKUP/COMPARE UPLIFT.EXE;3 UPLIFT.EXE;2
Compare a save set and a Files-11 file	$ BACKUP/COMPARE save-set-specifier file-spec
	$ BACKUP/COMPARE MTA0:1618FEB2.BCK/SELECT=[POUDRE]UPLIFT.PAS UPLIFT.PAS
Compare an image save set and Files-11 files	$ BACKUP/COMPARE/IMAGE save-set-specifier ddcu:
	$ BACKUP/COMPARE/IMAGE MTA0:12OCT.BCK DRA3:

Table 3-6 shows BACKUP command formats for a list operation, and some of the qualifiers associated with a list operation.

Table 3-6: List Operation Quick Reference Table

Command Action	Command Format Command Example
List the files in a save set at the terminal	$ BACKUP/LIST save-set-specifier
	$ BACKUP/LIST MTA0:1618FEB2.BCK
List the files in a save set, write to a file	$ BACKUP/LIST=file-spec save-set-specifier
	$ BACKUP/LIST=NEWLIST.LIS MTA0:1618FEB2.BCK
List the files in a save set in full format	$ BACKUP/LIST/FULL save-set-specifier
	$ BACKUP/LIST/FULL MTA0:1618FEB2.BCK
List selected files in a journal file	$ BACKUP/LIST/JOURNAL=journal-name/selection-qualifiers
	$ BACKUP/LIST/JOURNAL=SYS$MANAGER:INCBACKUP/SELECT=[LYKINS.WORK...]/SINCE=1-JAN-1982

CHAPTER 4

BAD BLOCK LOCATOR UTILITY (BAD)

The Bad Block Locator utility (BAD) determines and records the location of faulty blocks that cannot reliably store data. Table A-2 in Appendix A lists the block-addressable devices on which BAD can be used.

Usually, BAD tests block-structured volumes that have not been initialized. After BAD locates and records the bad blocks, you issue the DIGITAL Command Language (DCL) command INITIALIZE so that the operating system will allocate the faulty blocks to a special file. In this way, users are protected from accessing these faulty blocks for their files.

Section 4.1 below explains how BAD locates and records bad blocks; Section 4.2 explains how the INITIALIZE command allocates bad blocks. The remaining sections of this chapter describe how to invoke BAD, the BAD command line format and qualifiers, and the messages BAD can issue.

4.1 LOCATING AND RECORDING BAD BLOCKS

BAD locates bad blocks on a volume by testing whether the same data that is written into blocks can be read out. When it finds a bad block, BAD writes the address of that block into the bad block descriptor (described in Section 4.1.2).

4.1.1 Locating Bad Blocks

To test the blocks on a volume, BAD:

- Writes a test pattern onto each block
- Reads the contents of blocks into a buffer
- Compares the data in the buffer with the data it wrote into the blocks

If the data does not compare exactly, one or more blocks in the group of blocks are bad and cannot reliably store data. In this case, BAD will repeat the reading, writing, and comparing operations on each block in the group to determine the bad block(s).

4.1.2 Recording Bad Blocks

When BAD locates a bad block, it records the address of the block. Consecutive bad blocks are recorded as single entries. After it finishes testing the disk, BAD writes the addresses of the bad blocks into an area called the bad block descriptor.

4.1.2.1 Location of the Bad Block Descriptor - The location of the bad block descriptor depends on whether the volume is a last-track device. Last-track devices store bad block data on the last track of the disk.

The first half of the track is reserved for the Manufacturer's Detected Bad Sector File (MDBSF). The MDBSF stores the bad blocks discovered by the manufacturer when the device was originally formatted.

The second half of the track is reserved for the Software Detected Bad Sector File (SDBSF). The bad block descriptor is located here.

Last-track devices are:

- RK07, RL02 disk cartridges
- RM03/05 disk cartridges

Other devices (non-last-track devices) do not set aside the last track of the disk to store bad block information. Instead, BAD creates the bad block descriptor on the last good block of the disk. There must be at least one reliable block in the last 256 blocks of the volume for BAD to generate the bad block descriptor.

Non-last-track devices are:

- RP06 disk packs
- RX01/02 floppy diskettes
- TU58 DECtape II data cartridges

4.1.2.2 Format of the Bad Block Descriptor - If the volume is a last-track device, each bad block descriptor entry contains the cylinder, track, and sector addresses of the faulty block. The bad block descriptor can record a maximum of 126 entries.

On volumes that are not last-track, bad block descriptor entries contain the number of bad blocks minus 1 and bits 0 through 23 of the logical block number (LBN) of the faulty block or sequence of faulty blocks. A single entry can address one bad block or several contiguous bad blocks. The bad block descriptor on non-last-track devices can contain a maximum of 102 entries.

For both last-track and non-last-track devices, once the maximum number of entries is exceeded, BAD terminates with an error message.

4.2 ALLOCATING BAD BLOCKS

After you run BAD, the final step in processing bad block data is to issue the DCL command INITIALIZE. INITIALIZE changes the volume from unstructured format to Files-11 format and allocates the bad blocks found by BAD to a special file on the volume called [000000]BADBLK.SYS. Once they are allocated to BADBLK.SYS, the faulty blocks cannot be used by other files. For further information on Files-11 format and the INITIALIZE command, see the VAX/VMS Command Language User's Guide.

4.3 INVOKING AND TERMINATING BAD

When running BAD to test a device, keep in mind that:

- The device cannot be accessed by other programs
- The device cannot be mounted as a Files-11 volume
- The device is always purged by BAD's testing procedure; any information stored on the disk is destroyed

To ensure that the device is not accessed by any other programs, you must allocate the device with the DCL command ALLOCATE. See the VAX/VMS Command Language User's Guide for more information on the ALLOCATE command.

After you have allocated the device, you must give the DCL command MOUNT with the /FOREIGN qualifier. When the device is mounted foreign, the operating system does not recognize it as a Files-11 volume and BAD can execute.

There is no way to test the volume for bad blocks without destroying its contents. However, you can update the bad block descriptor without wiping out the volume by using the BAD qualifier /UPDATE. This qualifier is described in detail in Section 4.5.5.

To invoke BAD, enter the following command in response to the DCL prompt:

```
$ RUN SYS$SYSTEM:BAD
```

The utility responds with the prompt:

```
BAD>
```

You can now enter any BAD command string (Section 4.4). To return to DCL at any time, type CTRL/Z.

You can also invoke BAD by using the RSX-11M Monitor Console Routine (MCR) command:

```
MCR BAD [device-name:[/qualifier...]
```

The device name format is the same as described in Section 4.4.

4.4 BAD COMMAND STRING

The BAD command string has the following format:

```
BAD> device-name:[/qualifier...]
```

device-name

The device containing the volume on which BAD will be run. The device name has the form:

```
ddu
```

where

```
dd = 2- character alphabetic device code

  u = 1- or 2-digit octal device unit number
```

The colon (:) acts as the device-name terminator and must follow the device name. BAD does not recognize alphabetic controller designators. You must convert them to RSX-11M unit numbers when specifying devices. For information on conversion between VAX/VMS native mode unit numbers and compatibility mode unit numbers, see the explanation of mapping physical device names in the VAX-11/RSX-11M User's Guide.

/qualifier(s)

The BAD qualifiers that modify BAD operation. Multiple qualifiers are entered on the same command line; no separators are required. Section 4.5 discusses the BAD qualifiers in detail.

4.4.1 Running BAD Interactively from Your Terminal

The example below shows the sequence of commands that you should use when running BAD interactively from your terminal:

```
$ ALLOCATE DBA2:
_DBA2: ALLOCATED
$ MOUNT/FOREIGN DBA2:
%MOUNT-I-MOUNTED     mounted on _DBA2:
$ RUN SYS$SYSTEM:BAD
BAD>DB2:
BAD -- TOTAL NO. OF BAD BLOCKS = 2.
BAD> CTRL/Z
$
```

The ALLOCATE command requests the allocation of a specific disk drive, DBA2. The response from the ALLOCATE command indicates that the device was successfully allocated. The MOUNT/FOREIGN command mounts the disk volume as a foreign disk. The MOUNT command response indicates that DBA2 was successfully mounted. The RUN SYS$SYSTEM:BAD command invokes BAD. Specifying DB2 causes BAD to analyze each block on the disk volume and record the bad blocks. After BAD has tested all the blocks, it indicates that the number of bad blocks on DBA2 is 2. You exit from BAD by entering CTRL/Z in response to the BAD> prompt.

4.4.2 Running BAD from Command Procedures

You can invoke the BAD utility from a VAX/VMS command procedure. The following is a command procedure, named STEPS.COM, that invokes BAD and gives other DCL commands.

```
$ ALLOCATE DBB1:
$ MOUNT/FOREIGN DBB1:
$ RUN SYS$SYSTEM:BAD
DB21:/LI
$ DISMOUNT/NOUNLOAD DBB1:
$ INITIALIZE DBB1:
```

To call the command procedure, type the following in response to the DCL prompt:

```
$ @STEPS
```

The operating system executes the commands in the order they are given within the command procedure.

Note that because you are calling the command procedure from DCL, the default file type is COM and need not be specified. For a thorough discussion of command procedures, refer to the VAX/VMS Guide to Using Command Procedures.

BAD also allows you to use a command procedure to execute a series of BAD commands. The following example is a command procedure, named BADCMD.CMD, that contains the commands BAD is to execute. These commands are explained in Section 4.5.2.

```
DM2:/MAN
45
102
CTRL/Z
```

To call the command procedure, type:

```
$ MCR BAD

BAD> @BADCMDS
```

The default file type in this example is CMD because you are calling the command procedure from the MCR command interpreter.

BAD is invoked, performs the requested functions, and exits. Note that you can omit the file type when you call the command procedure. You can call up to three other command procedures from within one command procedure.

4.5 BAD QUALIFIERS

BAD provides five qualifiers which, when added to the command string, modify BAD operation. Table 4-1 lists the BAD qualifiers and gives a summary their functions.

Table 4-1: BAD Qualifiers

Qualifier	Notation	Function
List	/LI	Lists logical block numbers of bad blocks at your terminal
Manual	/MAN	Enters specific bad blocks to the bad block descriptor and tests the disk volume
Override	/OVR	Converts last-track devices to non-last-track devices
Retry	/RETRY	Enables the device driver to correct soft errors
Update	/UPD	Updates the bad block descriptor without testing the disk

The following sections describe each of these qualifiers in detail.

4.5.1 The List Qualifier

The List qualifier (/LI) causes all bad blocks to be listed by logical block number (LBN) on your terminal. Each time BAD encounters a faulty block, it writes the following message:

```
     BAD -- BAD BLOCK FOUND - LBN=  n
```

The value of n is the logical block number (LBN) of the block in decimal.

This qualifier is valid for all devices. If you do not specify the LIST qualifier, BAD will execute without listing the bad blocks at your terminal.

Example

```
     BAD> DB2:/LI

     BAD -- DB2:  BAD BLOCK FOUND - LBN= 20663
```

4.5.2 The Manual Qualifier

The Manual qualifier (/MAN) permits you to enter specific blocks to the bad block descriptor of an unformatted volume. You may want to use this qualifier to allocate specific blocks or a series of blocks so that they will not be used by other files.

When you use the Manual qualifier, BAD prompts for the logical block number of the blocks you want to enter:

```
     BAD> LBN(S)=
```

You can specify a single block or consecutive blocks in the form:

```
     lbn[:count]
```

The value of lbn is the logical block number in decimal and count is the number of consecutive blocks beginning at the specified LBN. You must use the colon (:) when specifying consecutive blocks and separate different LBNs or consecutive LBNs on a single line by a space, comma, or tab. Both lbn and count default to decimal unless they are preceded by the number sign (#) to indicate octal.

After you finish entering specific bad blocks, press CTRL/Z or ESC. BAD enters the blocks you have specified into the bad block descriptor and then tests the volume for bad blocks.

If you do not specify an LBN in response to the prompt but instead press RETURN, BAD lists the contents of the bad block descriptor in the format:

```
lbn:count
```

The value of lbn is the initial LBN of a possible sequence of bad blocks and count is the number of bad blocks in the sequence (in decimal). Single blocks are represented in the same format as a sequence of bad blocks.

Examples

1.

```
BAD>DB0:/MAN
BAD> LBN(S)= 45 CTRL/Z
BAD -- DB0:  TOTAL BAD BLOCKS= 2.
```

BAD enters the block represented by LBN 45 into the bad block descriptor, tests the disk for bad blocks, and reports the total number of bad blocks that it found at your terminal. (BAD does not include manually entered blocks in this total).

2.

```
BAD>DM2:/MAN
BAD> LBN(S)= 100:2,3 200:10.ESC
BAD -- DM2:  TOTAL BAD BLOCKS= 13.
```

BAD enters blocks 100, 101, 3, and 200 through 209 into the bad block descriptor.

3.

```
BAD>DM2:/MAN
BAD> LBN(S)= RET
000100:002
000003:001
000200:10
```

BAD lists all blocks in the bad block descriptor by logical block number and count.

4.5.3 The Override Qualifier

The Override qualifier (/OVR) causes BAD to ignore last-track information (the MDBSF and SDBSF, described in Section 4.1.2.1).

When specified, /OVR creates a bad block descriptor on the last good block before the last track of the disk, but does not generate a message at your terminal. If the last track does not contain a bad block descriptor, or if you suspect that the last track is faulty, use /OVR.

The Override qualifier converts last-track devices to non-last-track devices; thus, it is valid only on last-track devices.

4.5.4 The Retry Qualifier

The Retry qualifier (/RETRY) enables the device driver to correct soft errors. A soft error is a type of hardware error that causes good blocks to appear faulty. If /RETRY is not specified, BAD will prevent the device driver from correcting soft errors, and blocks mistakenly identified as bad will not be discovered.

4.5.5 The Update Qualifier

The Update qualifier (/UPD) enters additional blocks to the bad block descriptor without testing the volume. Use the Update qualifier when you want to update the bad block descriptor without destroying the contents of the volume.

The Update qualifier prompts for additional bad blocks in the same way as the Manual qualifier. When you have finished specifying the logical block numbers of the blocks you want entered to the bad block descriptor, press CTRL/Z or ESC. BAD will update the descriptor and exit.

If you do not specify an LBN in response to the prompt, but instead press RETURN, BAD lists the contents of the bad block descriptor in the format

```
     lbn:  count
```

as described in Section 4.5.2.

Example

```
     $ RUN SYS$SYSTEM:BAD
     BAD> DM1:/UPD
     BAD> LBN= 123
     BAD> CTRL/Z
```

The Update qualifier causes BAD to enter logical block number 123 into the bad block descriptor. Entering CTRL/Z returns control to DCL without testing the device.

4.6 BAD MESSAGES

The VAX/VMS System Messages and Recovery Procedures Manual lists the diagnostic messages generated by BAD, and provides explanations and suggested user actions for these messages.

CHAPTER 5

COMMAND DEFINITION UTILITY (SET COMMAND)

This chapter describes the Command Definition Utility. This facility enables you to modify the DIGITAL command language (DCL). The chapter consists of four parts:

1. An overview of a command language interpreter (CLI)
2. A discussion of how to modify DCL by using:
   - The command description language to define new commands
   - The DCL command SET COMMAND to process command descriptions
3. A description of command language interface routines that can be used by programs (images) to obtain information about the command strings that invoke them
4. A description of additional routines that allow a user program to obtain DCL-like parsing of command strings that it supplies

5.1 OVERVIEW

A command language interpreter (CLI) can be described as a set of routines that:

- Accept a user command string
- Parse the user command string according to command language tables, and create an internal representation of the user command string
- Invoke an image to perform the command
- Allow the image to request information from the CLI about the command string

Figure 5-1 illustrates the first three steps of this process.

When you supply a command string to a CLI, a set of command language parsing routines parse the command string and determine whether the syntax of the command string is valid. The parsing routines determine acceptable command syntax by examining your process's command language tables.

These tables contain data that describe commands and their attributes. This data is read by the parsing routines in order to obtain default values for command attributes that are not specified in your command string. The parsing routines are designed to prompt you for required command attribute information that is neither specified in your command string nor obtainable (by default) from data in the command language tables.

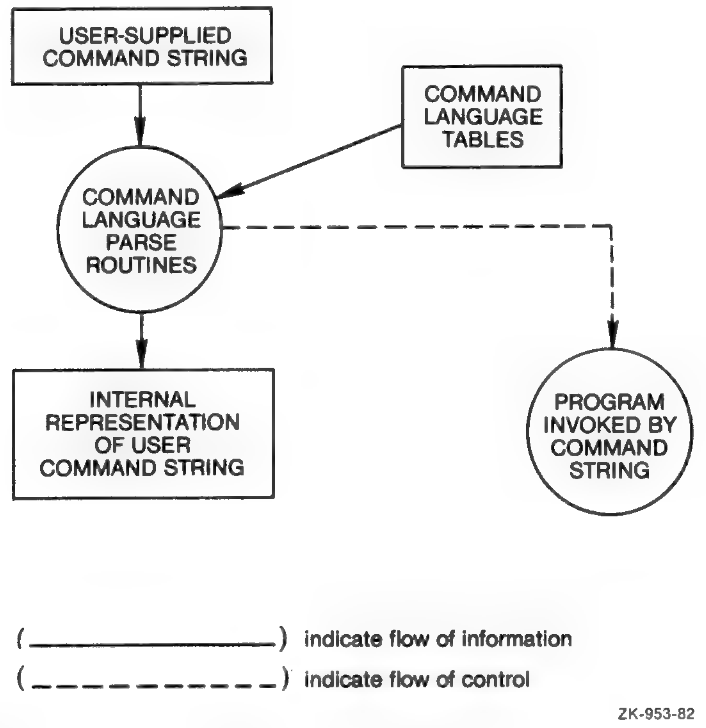

USER-SUPPLIED COMMAND STRING
COMMAND LANGUAGE TABLES
COMMAND LANGUAGE PARSE ROUTINES
INTERNAL REPRESENTATION OF USER COMMAND STRING
PROGRAM INVOKED BY COMMAND STRING
(————————) indicate flow of information
(— — — — — —) indicate flow of control
ZK-953-82

Figure 5-1: Supplying a Command String to a CLI

For example, the routines are designed to ask you for the name of a parameter that is required for a command verb, but not entered in your original command string.

The command language parsing routines create an internal representation of the command string. The CLI invokes the image requested by the command and then passes control to the image.

In the fourth step of this process (described below in Figure 5-2), the controlling image or program calls a set of command language interface routines in order to obtain information about the command string. The interface routines examine the internal representation of the command string and then return a status informing the program about the values or state of the command string.

Figure 5-2 illustrates this process.

Note that in Figure 5-2 the user-supplied command string and command language parsing routines (that were both included in Figure 5-1) are no longer present. Control of the process was transferred from the command language parsing routines to the user program. The controlling program is now able to obtain information from the internal representation of the command string that was originally supplied by the user-entered command string.

Section 5.1 has described the general features of a CLI. Sections 5.1.1, 5.1.2 and 5.1.3 describe each of these features in greater detail. In particular, the behavior of the DIGITAL Command Language (DCL) Interpreter is described.

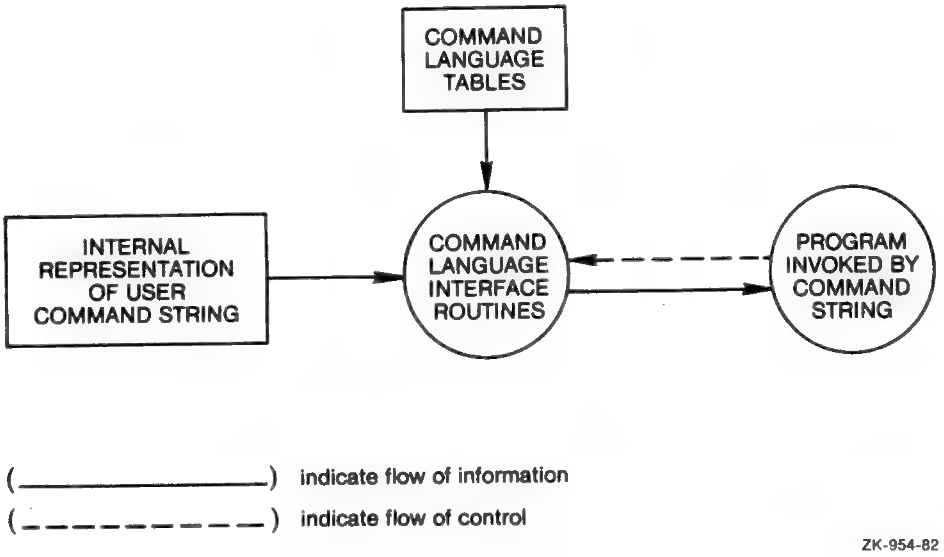

COMMAND LANGUAGE TABLES
INTERNAL REPRESENTATION OF USER COMMAND STRING
COMMAND LANGUAGE INTERFACE ROUTINES
PROGRAM INVOKED BY COMMAND STRING
(________) indicate flow of information
(- - - - - -) indicate flow of control
ZK-954-82

Figure 5-2: Obtaining Information About the Command String

5.1.1 Command Language Parsing Routines

When you supply a command string to the DCL interpreter, a set of DCL parsing routines examine the command string to determine its validity. The DCL parsing routines interpret this string by comparing it with the command descriptors contained by the DCL tables, which contain a description of the command verb and its attributes.

For example, if a user enters the command string:

```
     $ PRINT MYFILE/COPIES=2
```

the DCL parsing routines break the command string down into its components in order to determine the validity of the PRINT verb and the command string using that verb. If portions of the command string are missing, the DCL parsing routines examine the DCL tables to determine whether the unspecified portions of the command string can be supplied with default values from information already contained in the tables, or whether these values must be entered specifically by the user.

In the above command string, the PRINT command is followed by a parameter called MYFILE and a qualifier, /COPIES=2. In this case, the parameter MYFILE is the name of the file that you wish to print. The PRINT command requires that you specify the name of the file to be printed in your command string. If the parameter naming the file to be printed had not been obtained from the command string, the DCL parsing routines would request the file name from you by issuing the following prompt:

```
     $_File:
```

In this case, you would be required to enter the name of the file that you wish to print.

In certain cases, the DCL parsing routines will be able to obtain default values for you. For example, no queue name describing where the file was to be printed was specified in the above command string. In order to determine where MYFILE is to be printed, the parsing routines examine the DCL tables to see whether a default print queue has been defined.

The DCL tables can include the name of a specific print queue -- such as SYS$PRINT -- to be used when no specific queue is specified in the command string. In this case, the SYS$PRINT queue would be the default queue for the "PRINT" command. You can, however, override any default value by including an explicit value in the command string. For example, you could enter:

```
$ PRINT    MYFILE/COPIES=2   /QUEUE=LPB1
```

This string informs the DCL command PRINT that MYFILE is to be submitted to a queue called LPB1.

The /COPIES=x qualifier is also optional. If this qualifier is omitted from the command string, the parsing routines will examine the DCL tables to see whether they contain a default value for the number of copies of the file to be printed. In the case of the PRINT command, if the /COPIES=2 qualifier was not specified in the command string, only one copy of MYFILE would be printed.

After supplying all default values and determining that that the PRINT command string conforms to the syntax of DCL, the parsing routines create an internal representation of this command string and invoke the image associated with the PRINT command. Control is then passed from the parsing routines to the PRINT program (image).

5.1.2 Invoking the Program

The program (image) invoked by the PRINT command is now in a position to request information about its invoking command string. To obtain information from this command string, the program may call the set of command language interface routines. These routines are described in Section 5.1.3.

The program can be designed to call the interface routines to check the command string for the presence (or absence) of a certain parameter, such as the file name input parameter required for the PRINT command. It can also be designed to ask the routines to get the value of a certain qualifier, such as the /COPIES file qualifier for the PRINT command. The routines can first check the internal representation of the command string for the requested value. If no value had been explicitly specified in the command string, the routines then check the DCL tables to see whether a default value was included.

You can design the program to display a message, describing the completion of your command request. In the above PRINT command string, for example, the PRINT program signals the completion of the command request by displaying the following message on your terminal:

```
JOB 276 entered on queue SYS$PRINT
```

In order to carry out its function, the program relies on the command language interface routines to provide it with the information it needs from your command string and the DCL tables.

5.1.3 Command Language Interface Routines

The DCL interface routines enable an image, invoked by a command such as PRINT, to obtain information from DCL's internal representation of the command string. In the case of the DCL command PRINT, these interface routines are called to check the command string for the presence or absence of a parameter that is required as input for the PRINT command, and for the presence (or absence) of a qualifier that can be optionally specified. The interface routines also perform the function of getting the value of a qualifier from the command string or from the appropriate DCL tables and returning this value to the controlling image.

Section 5.1 described the features of a CLI in general and further illustrated these features in terms of one DCL command -- that is, the PRINT command. Section 5.2 describes how you can modify DCL by defining your own commands.

5.2 MODIFYING THE COMMAND LANGUAGE

You can use the Command Definition Utility to modify and extend the DCL command language. This utility enables you to define your own command verbs to execute user-written programs.

There are three ways to execute user-written programs. You can use:

- The DCL command RUN
- The Foreign Command facility
- The Command Definition Utility

The DCL command RUN allows you to execute a program image. It does not, however, enable you to pass additional information to that image.

The foreign command facility, on the other hand, enables you to pass additional information to the image. With the foreign command facility, however, the program must parse the remainder of the string in order to obtain the information it needs.

You can define a foreign command in the following way:

```
     $ command-name := $file-spec
```

command-name

The name of the foreign command you want to define.

file-spec

The fully qualified file specification of the executable image to be run when the command-name is invoked.

The following is an example of a command defined by the foreign command facility:

```
     $ DO := $USERDISK:[SMITH]TASK
```

Note that the dollar sign ($) prefix is required and must immediately precede the specification.

For details on using the foreign command facility, see the VAX-11 Run-Time Library Reference Manual and the VAX/VMS Command Language User's Guide.

The Command Definition Utility also enables you to pass information to an image. When using this utility, you can execute a command without knowing the file specification of its associated image.

To use the Command Defintion Utility, the following two steps are required:

1. Create a command description file to define a command (using any text editor)

2. Process the command description file (using the DCL command SET COMMAND)

The sections that follow discuss each of these steps in detail.

5.2.1 Creating a Command Description File

A command description file contains the information that defines a DCL command. A command description file:

- Defines verb, parameter, and qualifier characteristics

- Indicates the image to be invoked for each command

- Specifies alternate syntaxes for commands

You can use any text editor to create a command description file which has a default file type of CLD.

A command description file must be in the format of the command definition language. The command definition language is used to define the entities in the command string that will be requested by the image (by way of the DCL interface routines).

The command definition language is keyword-oriented. (A keyword is a word that the system recognizes.)

Table 5-1 lists the keywords that are recognized by the Command Definition Utility.

Table 5-1: Command Definition Utility Keywords

DEFINE VERB	QUALIFIER	PLACEMENT
DEFINE SYNTAX	REQUIRED	GLOBAL
IMAGE	VALUE	LOCAL
ROUTINE	LIST	POSITIONAL
PARAMETER	DEFAULT	SYNTAX
LABEL	BATCH	NOPARAMETERS
PROMPT	NEGATABLE	NOQUALIFIERS
MODULE	NONNEGATABLE	

It is important to note that keywords must not be abbreviated.

Figure 5-3 illustrates the keywords used in creating a simple command description file.

You can define any number of verbs in a single command description file. When you run the Command Definition Utility, these definitions are compiled and translated into command tables.

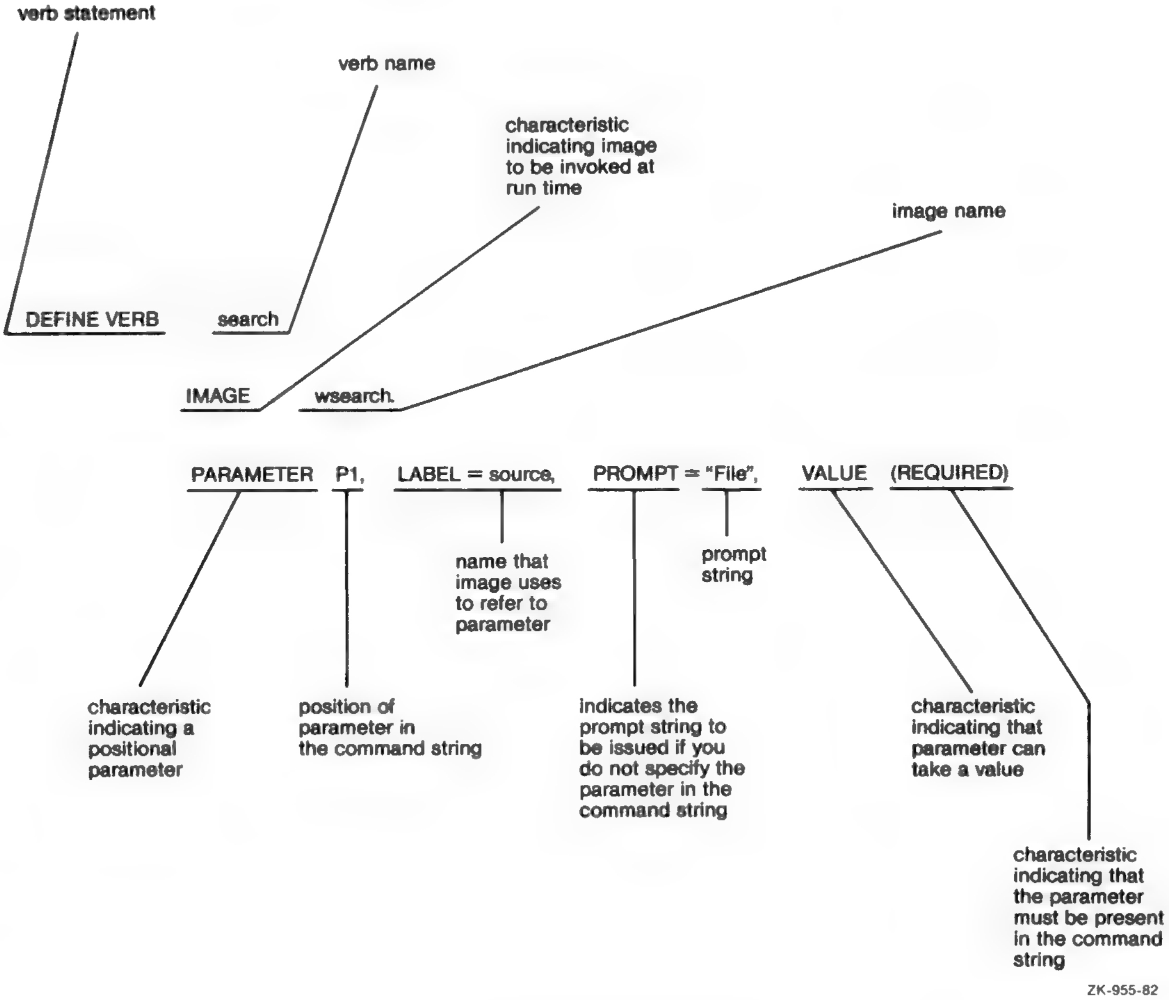

verb statement
verb name
characteristic indicating image to be invoked at run time
image name
DEFINE VERB
search
IMAGE
wsearch.
PARAMETER
P1,
LABEL = source,
PROMPT = "File",
VALUE
(REQUIRED)
name that image uses to refer to parameter
prompt string
characteristic indicating a positional parameter
position of parameter in the command string
indicates the prompt string to be issued if you do not specify the parameter in the command string
characteristic indicating that parameter can take a value
characteristic indicating that the parameter must be present in the command string
ZK-955-82

Figure 5-3: A Command Description File

5.2.1.1 Defining a Verb Using Verb Statements

Verb statements are used in the command description file. You use verb statements to specify different characteristics for the verb you are defining. The Command Definition Utility recognizes two verb statements: DEFINE VERB and DEFINE SYNTAX. The DEFINE SYNTAX statement is discussed in Section 5.2.1.3.

You use the DEFINE VERB statement to define a command. The format for this statement is:

```
DEFINE VERB verb name, verb characteristic, verb characteristic
```

When defining verbs, you must observe the following rules:

- The verb name is followed by a list of verb characteristics.
- All characteristics must be separated by commas or new lines.

- The exclamation point character (!) causes all following characters on the line to be treated as comments.

- Any definition can be broken into multiple lines by starting a new line following a comma; no continuation character is necessary.

- Keywords in the command definition language are not allowed to be abbreviated; all keywords must be spelled out completely.

An example of a simple verb definition is:

```
DEFINE VERB erase
```

This definition tells the command language interpreter that erase is a valid verb and that it takes no parameters or qualifiers. It also tells the command language interpreter to invoke the image ERASE because no explicit image name is specified. When you specify an image (for example, the image ERASE) you need specify only the file name, as opposed to the entire file specification. The command language interpreter supplies a default value of SYS$SYSTEM for the directory field and a default of EXE for the file type field of the file specification.

5.2.1.2 Specifying Characteristics for the Verb - Verb characteristics describe the verb. They can appear in any order. However, since the command language interpreter makes only one pass, no forward references to symbols are allowed. Verb characteristics perform the same function for both verb statements (DEFINE VERB and DEFINE SYNTAX). All verb characteristics are optional.

Figure 5-4 contains a list of the characteristics used for defining your verbs.

```
IMAGE image-name


ROUTINE routine-name


PARAMETER position, parameter-characteristic,
parameter-characteristic,...

        Parameter-Characteristics:
                        LABEL=name
                        PROMPT=string
                        SYNTAX=name
                        VALUE=[(characteristic, characteristic,...)]

                                Value Characteristics:
                                        REQUIRED
                                        DEFAULT=string
                                        LIST
```

Figure 5-4: Verb Characteristics

```
QUALIFIER qualifier-name, qualifier-characteristic,
qualifier-characteristic,...

        Qualifier-Characteristics:
                        LABEL=name
                        DEFAULT
                        BATCH
                        NONNEGATABLE
                        NEGATABLE
                        PLACEMENT=GLOBAL
                                  LOCAL
                                  POSITIONAL
                        SYNTAX=name
                        VALUE[(charac, charac,...)]
                                REQUIRED
                                DEFAULT=string
                                LIST

NOPARAMETERS  (applies to DEFINE SYNTAX only)

NOQUALIFIERS  (applies to DEFINE SYNTAX only)
```

Figure 5-4 (Cont.): Verb Characteristics

Each of the characteristics used for defining a verb is discussed below in detail.

IMAGE image-name

The name of the image that DCL invokes when you specify the command. If this attribute is not specified, verb-name is used as the image-name. At run time, DCL searches for an image whose default file name is the same as the verb name and whose whose default device name and file type are SYS$SYSTEM: and EXE, respectively.

ROUTINE routine-name

The name of a routine used in processing application supplied command strings. The use of these routines is discussed in Section 5.4

PARAMETER position, parameter-characteristic, parameter-characteristic

A parameter that can be specified in the command string. Each parameter in the command string is separated from the next by a space. You can specify up to eight parameters for a single command. Each parameter number must be explicitly specified. The parameters must be numbered consecutively from P1 to P8.

position

The position of the parameter in the command string. Parameter position must be in the form Pn, where n is the position of the parameter. You can specify up to eight parameters for a single command. Position is a required field and must immediately follow the PARAMETER keyword.

Parameter-Characteristics:

LABEL=name

The label name to be used when requesting information about the item at run time. The name can contain letters, numbers, or underscores (_), and can be up to 31 characters in length. If you do not specify a label name, the parameter as given in position P1 through P8 is used as the label name.

PROMPT=string

A prompt string issued if you do not specify a value for a required parameter in the command string. If the prompt string contains characters other than letters, numbers, or underscores (_), it must be enclosed in quotation marks ("). If you do not specify the prompt string and the required parameter is missing, DCL will issue an error message. (See REQUIRED below.) By default, the Command Definition Utility converts all characters to uppercase. If you wish to override this default, you must enclose the characters in quotation marks.

VALUE (characteristic,characteristic,...)

An optional characteristic of the parameter. It can be followed by a list of VALUE characteristics enclosed in parentheses. The list of characteristics appears below.

Value-Characteristics:

REQUIRED

Indicates that the parameter is required. If you do not specify the parameter in the command string, and a prompt string has been specified, you receive a prompt from the DCL CLI (except when using CLI$DCL_PARSE as described in Section 5.4.1.1). If REQUIRED is not specified, then the parameter is optional and no prompting is done. The REQUIRED characteristic and the DEFAULT=string characteristic are mutually exclusive parameter characteristics.

DEFAULT=string

The default value to be used in the absence of an explicit value. The DEFAULT=string characteristic and the REQUIRED characteristic are mutually exclusive parameter characteristics.

LIST

Indicates that a list of values separated by commas or plus signs (+) can be specified for the parameter.

QUALIFIER name, qualifier-characteristic, qualifier-characterisitc,...

Defines how a specified qualifier is to appear in the command string. A qualifier is a characteristic represented by a qualifier name and preceded by a slash (/). It can appear anywhere in the command string. The name is required and must follow the QUALIFIER keyword.

Qualifier-Characteristics:

LABEL=name

The qualifier name to be used when requesting information about the item at run time. If you do not specify a label name, the qualifier name is used as the label name.

DEFAULT

Indicates that the qualifier is present by default.

BATCH

Indicates that the qualifier is present by default if the command is used in a batch job.

NEGATABLE

"NO" can be used on the qualifier to indicate that the qualifier is not present. This is the default; thus, it need not be specified.

NONNEGATABLE

Indicates that you cannot negate the qualifier's presence by adding "NO" to the qualifier name. By default, "NO" can be used on any qualifier to indicate that the qualifier is not present.

```
PLACEMENT=GLOBAL
          LOCAL
          POSITIONAL
```

Indicates where the qualifier can appear on the command line. GLOBAL means that the qualifier can appear anywhere. The qualifier then applies to the entire command. LOCAL means that the qualifier can appear only after a parameter, and applies only to that parameter. POSITIONAL means that if the qualifier appears after a parameter, it applies only to that parameter; if it appears after the verb, it applies to all parameters. PLACEMENT=GLOBAL is the default.

SYNTAX=name

Specifies an alternate verb definition to be invoked when this qualifier is present. This is useful for commands that invoke different images depending upon the particular qualifiers that are present.

VALUE [(characteristics,characteristics,...)]

An optional characteristic of the qualifier. It can be followed by a list of characteristics. The list of characteristics appear below. Note that no prompting is done by DCL for the REQUIRED characteristic for qualifiers.

Value-Characteristics:

REQUIRED

Indicates that the qualifier must have an explicitly specified value. Unlike the parameter characteristic, which can prompt for a required value (when no value is specified), no prompting is allowed for a qualifier characteristic. Therefore, you must specify a value for a qualifier characteristic that is required. The REQUIRED characteristic and the DEFAULT=string characteristic are mutually exclusive qualifier characteristics.

DEFAULT=string

The default value to be used in the absence of an explicit value. The DEFAULT=string characteristic and the REQUIRED characteristic are mutually exclusive qualifier characteristics.

LIST

Indicates that a list of values separated by commas can be specified for the qualifier. Note that a plus sign (+) cannot be used to separate a list of values.

Table 5-2 illustrates three sample command description files, varying in degrees of complexity.

Table 5-2: Sample Command Description Files

How to define a command that:	Using a text editor, create a file containing a DEFINE VERB statement and the name of an image to be invoked. Return to DCL and invoke the Command Definition Utility by typing: SET COMMAND filename. The command description file would look like this:
Has no parameters or qualifiers:	DEFINE VERB simple IMAGE simple
Has one or more parameters:	DEFINE VERB scatter IMAGE scatter PARAMETER P1,LABEL=sign,PROMPT="yes?",VALUE(REQUIRED) PARAMETER P2,LABEL=arrow
Has one or more qualifiers:	DEFINE VERB seek IMAGE search QUALIFIER SEND,LABEL=under,DEFAULT

5.2.1.3 The DEFINE SYNTAX Statement and Syntax Lists - The DEFINE SYNTAX statement is similar to the DEFINE VERB statement in that it allows you to:

- Choose the image invoked to execute a command
- Include qualifiers and parameters in the command string

The DEFINE SYNTAX statement differs from the DEFINE VERB statement in the following way. You use the DEFINE SYNTAX statement to define an alternate command syntax. This alternate command syntax is called a syntax list and is a collection of verb characteristics grouped together under a given name. Syntax lists change or replace the syntax of a command when you specify a certain qualifier or parameter in the command string. A syntax list has the following format:

```
DEFINE SYNTAX syntax-name, verb-characteristic, verb-characteristic,...
```

You can use the verb characteristics listed below with a syntax list:

- IMAGE image-name
- PARAMETER position, param charac, param charac,...
- QUALIFIER name, qualif charac, qualif charac,...

- NOPARAMETERS
- NOQUALIFIERS

NOTE

> The syntax list must be declared before it is used in a verb definition with the SYNTAX=name keyword.

The characteristics NOPARAMETERS and NOQUALIFIERS specify that all parameters or qualifiers previously defined in the command description file are now disallowed.

Table 5-3 illustrates some syntax list conditions and results.

Table 5-3: Syntax List Conditions and Results

Syntax list contains:	Result:
An image	This image overrides the previous image.
A routine	This routine overrides the previous routine.
Any parameter	All parameter definitions in the syntax list override all previous definitions.
Any qualifier	All qualifier definitions in the syntax list override all previous definitions.
NOPARAMETERS or NOQUALIFIERS keyword, or both	No parameters or no qualifiers, or both, are permitted, regardless of previous definition.

Example 5-1 illustrates a command description file containing syntax lists in which both the SOS and SLP editor images can be used to replace the EDT editor image.

```
DEFINE SYNTAX SOS
          IMAGE SOS
          QUALIFIER NUM

DEFINE SYNTAX SLP
          IMAGE SLP
          QUALIFIER AUDIT

DEFINE VERB EDIT
          IMAGE EDT
          QUALIFIER SOS, SYNTAX=SOS
          QUALIFIER SLP, SYNTAX=SLP
```

Example 5-1: EDIT.CLD

Note that this sample command description file, describing the EDT, SOS, and SLP editors is only an example of how to use the DEFINE

SYNTAX statement and should not be construed as the CLD file that VAX/VMS uses to invoke these editors.

When you specify either a parameter or qualifier directing the command language interpreter to use a given syntax list, all the definitions in the syntax list override any previous corresponding definitions.

If you specify the /SOS qualifier with the EDIT command, you invoke the SOS editor instead of the default editor (EDT). Syntax redefinition is done from left to right because parsing of the line is done from left to right. This means that when you specify two qualifiers that invoke different syntax lists, the leftmost qualifier takes precedence (since it is parsed first). For example, if you type EDIT/SOS/NUM, you invoke the SOS editor with the NUM qualifier.

Parameters and qualifiers placed before the syntax switching entity are parsed according to the original verb definition. Parameters and qualifiers placed after the syntax switching entity are parsed according to the new verb definition.

5.2.1.4 The MODULE Keyword and Module Names - Unlike the keywords discussed above, which described verb characteristics, the MODULE keyword is used to describe the name of the parsing tables to be used when processing application-supplied command strings (see Section 5.4).

The MODULE keyword has the format:

```
     MODULE module-name
```

module-name

Name of the parsing tables that you create when using the /OBJECT command qualifier with SET COMMAND (see Section 5.2.2).

If no module name is specified, the Command Definition Utility will use the (specified) object file name as the default module name. If no object file is explicitly specified, then it will use the name of the first input file as the module name.

5.2.2 Processing the Command Description File

Command description files must first be translated into command tables that can be read by the CLI before the commands defined in them can be executed. To perform this translation, you must use the DCL command SET COMMAND.

When you enter the DCL command SET COMMAND, followed by the file specification of the command description file (default file type CLD) to be processed, DCL invokes the Command Definition Utility to create a new set of command tables. You can then begin using the newly defined command because DCL will now be able to recognize it.

By using the optional command qualifiers, you can create and modify command definitions, and save them for repeated use.

Format

```
SET COMMAND file-spec[,...]
```

Command Qualifiers	**Defaults**
/[NO]DELETE=(verb[,...])	/NODELETE
/[NO]LIST[=file-spec]	/NOLIST
/[NO]OBJECT[=file-spec]	/NOOBJECT
/[NO]OUTPUT[=file-spec]	(see text)
/[NO]TABLES[=file-spec]	(see text)

Prompts

File: file-spec

Specifies the name of one or more command description files. The default file type is CLD.

You can specify more than one command description file in the command string; to do so, separate file names with a comma. Wild card characters are permitted in the file specifications.

Command Qualifiers

/[NO]DELETE=(verb[,...])

Specifies the verb or verbs to be deleted from the command tables before processing any definition files. If you specify more than one verb, separate them with commas and enclose the list in parentheses. By default, no verbs are deleted.

/[NO]LIST[=file-spec]

Specifies that a listing that reflects the input is to be created. If no file name is specified, the file name will default to the name of the first input file. The default file type is LIS. By default, no listing file is created.

The listing contains error messages and is similar to a compiler listing.

/[NO]OBJECT[=file-spec]

Specifies that an object file containing command tables is to be created. If no file name is specified, it will default to the name of the first input file. The default file type is OBJ. By default, no object file is created.

The object file can be linked with other object files and used in conjunction with the CLI$DCLPARSE routine (see Section 5.4.1.1). Do not use the /OBJECT qualifier with either the /TABLES or the /OUTPUT qualifiers. The functions of these qualifiers are mutually exclusive.

/OUTPUT

Specifies that the edited command tables are to be placed in your process. This is the default; thus, it need not be specified.

The /OUTPUT and /OBJECT qualifiers are mutually exclusive.

/OUTPUT=file-spec

Specifies that the edited command tables are to be written to the file that you specify, rather than into the command tables in your current process. If you do not include a file name as part of the file specification for the output file, it will default to the name of the file given by the /TABLES =file-spec command qualifier. The default file type is EXE.

Note that when you use this qualifier, you must also use the /TABLES=file-spec qualifier to provide the input tables

/NOOUTPUT

Specifies that the edited command tables are not to be written anywhere.

/TABLES

Specifies that the command tables in the current process are to be edited. This is the default; thus, it need not be specified.

The /TABLES and /OBJECT qualifiers are mutually exclusive.

/TABLES=filespec

Specifies that the given command tables file will be edited instead of the current process command tables. The default file type is EXE.

/NOTABLES

Specifies that no input tables are to be used.

Table 5-4 illustrates various SET COMMAND qualifier specifications and their results.

Table 5-4: SET COMMAND Qualifier Specifications and Results

Specification	Result
$ SET COMMAND TEST	The TEST.CLD file is processed and the results are added to your process's command tables
$ SET COMMAND/TABLES TEST	The TEST.CLD file is processed and the results are added to your process's command tables (the result is the same as above)
$ SET COMMAND/TABLES=A TEST	The TEST.CLD file is processed, added to the command tables A.EXE, and inserted into your process
$ SET COMMAND/TABLES=A	The tables contained in a file named A.EXE are loaded into your process

(continued on next page)

Table 5-4 (Cont.): SET COMMAND Qualifier Specifications and Results

Specification	Result
$ SET COMMAND/OUTPUT TEST	The TEST.CLD file is processed and the results are added to your process's command tables (the result is the same as the first two cases)
$ SET COMMAND/OUTPUT=A - /TABLES=A TEST	The TEST.CLD file is processed, added to the command tables A.EXE, and the results are written to a new file named A.EXE
$ SET COMMAND/NOOUTPUT TEST	The TEST.CLD file is processed and the result is not used
$ SET COMMAND/OBJECT TEST	The TEST.CLD file is processed and the results are written as an object module to a file named TEST.OBJ
$ SET COMMAND/OBJECT=A TEST	The TEST.CLD file is processed and the results are written as an object module to a file named A.OBJ
$ SET COMMAND/LIST TEST	The TEST.CLD file is processed, the results are added to your process's command tables, and a listing file named TEST.LIS is created
$ SET COMMAND/LIST=A TEST	The TEST.CLD file is processed, the results are added to your process's command tables, and a listing file named A.LIS is created
$ SET COMMAND/DELETE=DO TEST	The verb DO is deleted from the current command tables before the TEST.CLD file is processed, and its results added to your process's command tables

By default, DCL uses the command tables contained in a file named SYS$LIBRARY:DCLTABLES.EXE for the initial process command tables when you log in to VAX/VMS.

5.3 COMMAND LANGUAGE INTERFACE ROUTINES

An image invoked by DCL as a result of a command defined by using the Command Definition Utility may want information about either the values or state of the command string that invoked it. You can use the Command Language Interface Routines to retrieve this information.

The two Command Language Routines that you use to obtain this information are CLI$PRESENT and CLI$GET_VALUE. You can call the

CLI$PRESENT routine to determine if an entity is present in the command string. You call the CLI$GET_VALUE routine to get the value of the next entity in the command string.

Table 5-5 demonstrates some of the uses of the CLI$PRESENT and CLI$GET_VALUE interface routines. It provides a list of conditions and results. The first three cases apply only to CLI$GET_VALUE.

Table 5-5: Command Language Interface Conditions and Results

Condition	Result
If the value is explicit in the command string,	Then it is returned via the string descriptor.
If the value is not present but has a default value,	Then that default value is returned via the string descriptor.
If the value is not present and has no default value,	Then the null string is returned via the string descriptor.
If a local or positional qualifier value is requested (for example, the parameter on which the qualifier is placed),	Then the qualifier is returned only if it appears on the most recently requested parameter value.
If the qualifier is global,	Then the order in which the qualifier is retrieved is irrelevant.
If a local qualifier is not present on a parameter,	Then the global qualifier (on the verb) is used.

5.3.1 CLI$PRESENT

This routine is called to determine whether a certain parameter or qualifier was present in the command string. The format for this routine is as follows:

```
ret-status = CLI$PRESENT(name)
```

name

This argument is the address of a string descriptor of the qualifier or parameter name. The name can be either the actual parameter or qualifier name or the string that was specified by the label characteristic. If LABEL was used, then you must specify the label name; otherwise, you must specify the name of the parameter or qualifier.

ret-status

The possible status conditions are described as follows:

TRUE	
CLI$_PRESENT	Value explicitly given
CLI$_DEFAULTED	Value defaulted present

FALSE

CLI$_ABSENT	Value not present and there is no default value
CLI$_NEGATED	Value explicitly negated (/NO)

If the entity that you specify does not exist, CLI$PRESENT will signal a syntax error (via the signalling mechanism described in the VAX-11 Run-Time Library Reference Manual).

5.3.2 CLI$GET_VALUE

This routine retrieves the value associated with a specified qualifier or parameter. It has the following format:

```
ret-status = CLI$GETVALUE (name, retbuf)
```

name

The first argument is the address of a string descriptor of the qualifier or parameter name. The name can be either the actual parameter or qualifier name or the string that was specified by the label characteristic. If LABEL was used, then you must specify the label name; otherwise, you must specify the name of the parameter or qualifier.

retbuf

The second argument is the address of a character string descriptor (the descriptor of the buffer) to receive the string value. The string is returned using the STR$COPYDX VAX-11 Run-Time Library routine.

ret-status

The possible status conditions are described as follows:

TRUE

CLI$_CONCAT	Value was concatenated to the next value with a plus sign (+)
SS$_NORMAL	Value was delimited from the next value by a comma, or there was only one value, or the value was the last value in the list

FALSE

CLI$_ABSENT	Value was not present or the last value in the list has already been received
Other Errors (Errors in specification of return descriptor or errors that occurred while trying to copy the results using that descriptor)	Various errors returned from the STR$COPY_DX routine. For a list of these errors, see the VAX-11 Run-Time Library Reference Manual

If you ask for an entity that does not exist, CLI$PRESENT will signal a syntax error (via the signalling mechanism described in the VAX-11 Run-Time Library Reference Manual).

For lists of values, this routine can be called iteratively to obtain the next value in the list until the routine status is false (end of list).

The following special label names describe special strings that can be retrieved using CLI$GET_VALUE:

$VERB Describes the verb in the command string (the first four letters of the spelling as defined, instead of the string that was actually typed)

$LINE Describes the entire command string as typed, including the verb, and can be used to retrieve the entire string for special parsing

5.3.3 Using the Command Language Interface Routines

You can design the program, which will be invoked by your newly defined command verb, to include one or both of these command language interface routines. This section includes an example of a sample program that uses each of the command language interface routines described above. It also includes an example of a command description file that is designed to invoke the sample program.

Example 5-2 illustrates a simple application program called SAMPLE.BAS, which uses the CLI$PRESENT and CLI$GET_VALUE routines.

```
1     EXTERNAL INTEGER FUNCTION CLI$PRESENT,CLI$GET_VALUE

10    IF CLI$PRESENT('EDIT') AND 1%
      THEN
          PRINT '/EDIT IS PRESENT',A$

20    IF CLI$PRESENT('FILESPEC') AND 1%
      THEN
          CALL CLI$GET_VALUE('FILESPEC',A$)
          PRINT 'FILESPEC = ',A$

30    END
```

Example 5-2: SAMPLE.BAS

This source program must first be compiled and linked before it can be invoked by a command verb. When you have finished compiling and linking the source program you will have created a file called SAMPLE.EXE, which contains an executable image. This image can then be invoked by a command verb.

Example 5-3 contains a simple command description file that is designed to invoke the SAMPLE image.

```
DEFINE VERB SAMPLE
       IMAGE  USERDISK:[MYDIR]SAMPLE
       PARAMETER  P1,LABEL=FILESPEC
       QUALIFIER  EDIT
```

Example 5-3: SAMPLE.CLD

To process this command description file, you use the DCL command SET COMMAND:

```
$ SET COMMAND SAMPLE
```

This command string invokes the Command Definition Utility, which processes the command description file (SAMPLE.CLD) and adds the verb SAMPLE to the command tables into your process. You can now use the SAMPLE command, which will invoke the SAMPLE image. You enter the following command string:

```
$ SAMPLE
```

The SAMPLE command verb is interpreted and processed in the same way as the DIGITAL-supplied DCL commands, whose tables were originally included in your process.

You may include in the command string any parameters and qualifiers defined for the SAMPLE command verb. For example, if you wish to include the file-spec parameter, you can the enter the following command string:

```
$ SAMPLE MYFILE
```

In this case, you will receive the following display on your screen:

```
FILE-SPEC = MYFILE
```

You can also include the /EDIT qualifier in the command string. For example, you can enter the following command string:

```
$ SAMPLE MYFILE/EDIT
```

In this case, you will receive the following display on your screen:

```
/EDIT IS PRESENT

FILE-SPEC = MYFILE
```

If you include a qualifier that is not accepted by the command verb, you will receive a DCL error message. If, for example, you enter:

```
$ SAMPLE MYFILE/UPDATE
```

You will receive the following message:

```
%DCL-W-IKEYW,unrecognized qualifier keyword
  \UPDATE\
```

If you include two or more parameters in the command string for a verb that was defined to accept only one parameter, you will also receive an error message. For example, if you enter:

```
$ SAMPLE MYFILE INFILE
```

You will receive the following display on your screen:

```
%DCL-W-MAXPARM,maximum parameter count exceded
  \INFILE\
```

The above two DCL error messages inform you that neither the /UPDATE qualifier nor multiple parameters are accepted by the SAMPLE command verb.

Section 5.3 has illustrated the way in which you can use the Command Definition Utility to extend the DCL command language to include an additional command, such as SAMPLE. Section 5.4 demonstrates another use of the Command Definition Utility.

5.4 PROCESSING APPLICATION-SUPPLIED COMMANDS

In the previous sections, the DCL command SET COMMAND was used to create new DCL-like commands, thus modifying the DCL command language. In these cases, the newly defined commands were treated as extended DCL commands and processed as such.

The Command Definition Utility can also be used to process commands supplied by an application program. The DCL command SET COMMAND can create an object file from the command description file that it processes. This object module can then be linked with your application program. In this use of the Command Definition Utility, the application program supplies the command strings to be processed. These command strings will have a valid meaning only for your application program.

Figure 5-5 illustrates the manner in which an application program supplies a command string to the DCL parse routines, which in turn create an internal representation of that command string. The string is then examined by the command language interface routines and requested information is passed back to the original program.

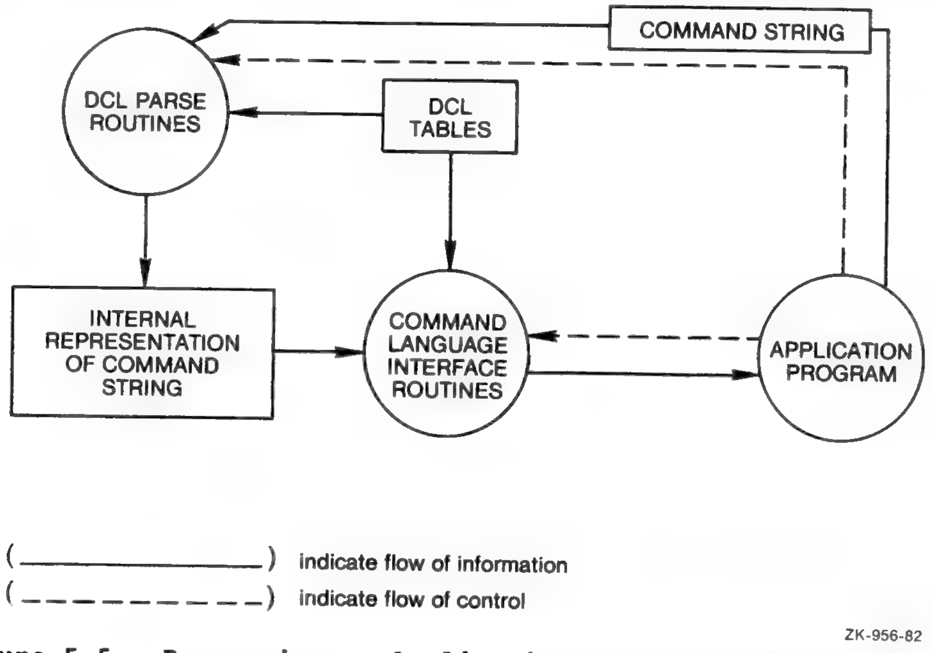

COMMAND STRING
DCL PARSE ROUTINES
DCL TABLES
INTERNAL REPRESENTATION OF COMMAND STRING
COMMAND LANGUAGE INTERFACE ROUTINES
APPLICATION PROGRAM
(————————) indicate flow of information
(– – – – – – –) indicate flow of control
ZK-956-82

Figure 5-5: Processing an Application-Supplied Command String

In the previous sections, example command description files contained the name of an image to be invoked by the command verb. Command description files designed for application-supplied command string processing, however, must include a routine name for each command verb. An example of a command description file that contains routine names (as opposed to image names) is provided below in Example 5-4.

The processing of application-supplied command strings requires the use of two additional command language interface routines that were not discussed in Section 5.3. The following sections describe these two routines and include an example of how they are used in processing application-supplied command strings.

5.4.1 Routines for Processing Application-Supplied Command Strings

There are two command language interface routines that you can use to process application-supplied command strings. These routines, CLI$DCL_PARSE and CLI$DISPATCH, can be used in conjunction with the routines discussed in Section 5.3.

5.4.1.1 CLI$DCL_PARSE - This routine is called to parse a command string. It parses an application-supplied command string (passed as a string descriptor in argument 1) as a DCL command, according to the command language definition tables (passed in argument 2). The format for this is as follows:

```
ret-status = CLI$DCL_PARSE (command string, tables)
```

command string

> The first argument is the address of a string descriptor specifying the command string to be parsed

tables

> The second argument is an external reference to a symbol whose name is the module name of the parsing tables that you created when using the /OBJECT qualifier with the DCL command SET COMMAND. For a description of how the module name is determined, see Section 5.2.1.4.

ret-status

> The possible status conditions can fall into one of two categories:

TRUE

Command string was successfully parsed

FALSE

Command string was not successfully parsed and an error has been signalled (via the signalling mechanism described in the VAX-11 Run-Time Library Reference Manual).

The CLI$DCL_PARSE routine ignores comments (delimited by exclamation marks) in the command string. This routine does not perform symbol substitution and is not able to handle command continuation.

You can retrieve the results of this parse by using the CLI$PRESENT, CLI$GET_VALUE, and CLI$DISPATCH command language interface routines.

5.4.1.2 CLI$DISPATCH - This routine invokes a routine that corresponds to the verb most recently parsed by CLI$DCL_PARSE. The format for this routine is as follows:

```
ret-status = CLI$DISPATCH()
```

This routine accepts no arguments.

ret-status

The status returned by your routine.

The CLI$DISPATCH routine can be called after the CLI$DCL_PARSE routine has successfully parsed a command string.

If no routine had been specified in the command description file, then no action will be performed by CLI$DISPATCH. In this case, the status condition will be successful (ret-status = SS$_NORMAL).

5.4.2 Processing Command Strings in a User Application

When you design a command description file for processing application-supplied command strings, you must include the name of a routine for each verb. These routines can then be called by your program when it processes your application-supplied command strings.

EXAMPLE 5-4 illustrates a command description file, called TEST.CLD, which contains the names ot three verbs: SEND, SEARCH, and EXIT. Each verb invokes its own routine.

```
MODULE TEST_TABLES

DEFINE VERB SEND
        ROUTINE SEND_COMMAND
        PARAMETER P1, LABEL = FILESPEC
        QUALIFIER EDIT

DEFINE VERB SEARCH
        ROUTINE SEARCH_COMMAND
        PARAMETER P1, LABEL = SEARCH_STRING

DEFINE VERB EXIT
        ROUTINE EXIT_COMMAND
```

Example 5-4: TEST.CLD

5.4.2.1 Designing the User Application - You can design a program to invoke each of the routines, or any single routine, contained in the above command description file.

When you invoke the Command Definition Utility to process this file, you should include the /OBJECT qualifier as part of the SET COMMAND string to indicate that you wish your newly created command tables to be written as an object module to a file named TEST.OBJ.

You can then link this object module (TEST.OBJ) with an object module that was assembled or compiled from your user source program, to create an executable image file (.EXE).

Example 5-5 contains a sample user program, entitled USEREXAMP.BAS, which is written in BASIC. This program is designed to use the two command language interface routines provided by the Command Definition Utility to invoke the routines associated with each of the verbs in your command description file, shown in Example 5-4.

```
10   SUB SENDCOMMAND
     EXTERNAL INTEGER FUNCTION CLI$PRESENT,CLI$GET_VALUE

     PRINT 'SEND COMMAND'
     PRINT ''

20   IF CLI$PRESENT ('EDIT') AND 1%
     THEN
         PRINT '/EDIT IS PRESENT'

30   IF CLI$PRESENT ('FILESPEC') AND 1%
     THEN
         CALL CLI$GET_VALUE ('FILESPEC',A$)
         PRINT 'FILESPEC = ',A$

90   SUBEND


100 SUB SEARCH_COMMAND
     EXTERNAL INTEGER FUNCTION CLI$PRESENT,CLI$GET_VALUE

     PRINT 'SEARCH COMMAND'
     PRINT ''

110 IF CLI$PRESENT('SEARCH_STRING') AND 1%
     THEN
         CALL CLI$GET_VALUE('SEARCH_STRING',A$)
         PRINT 'SEARCH_STRING = ',A$

190 SUBEND


200 SUB EXIT_COMMAND
     EXTERNAL INTEGER FUNCTION SYS$EXIT

     CALL SYS$EXIT(1%)

290 SUBEND


1    EXTERNAL INTEGER FUNCTION CLI$DCL_PARSE,CLI$DISPATCH,LIB$GET_INPUT
     EXTERNAL INTEGER FUNCTION SEND_COMMAND,SEARCH_COMMAND,EXIT COMMAND
     EXTERNAL INTEGER TEST_TABLES

2    CALL LIB$GET_INPUT(A$,'TEST> ')
     IF NOT CLI$DCL_PARSE(A$,TEST_TABLES) AND 1%
     THEN
         GOTO 2

3    PRINT ''
     CALL CLI$DISPATCH
     PRINT ''
     GOTO 2
     END
```

Example 5-5: USEREXAMP.BAS

This source program, which includes the CLI$DCL_PARSE and CLI$DISPATCH routines, must first be compiled before it can be linked with an object module created from the SET COMMAND/OBJECT command. To compile this program, you invoke the VAX-11 BASIC compiler with the DCL COMMAND:

```
$ BASIC USEREXAMP
```

You now have a USEREXAMP.OBJ file in addition to the original USEREXAMP.BAS source file. The USEREXAMP.OBJ file can be linked with the TEST.OBJ file, which is created with the DCL command:

```
$ SET COMMAND/OBJECT TEST
```

You then link the file containing the TEST object module (TEST.OBJ) with the file containing the USEREXAMP object module (USEREXAMP.OBJ) by issuing the DCL command:

```
$ LINK USEREXAMP,TEST
```

You now have a file containing an executable image (USEREXAMP.EXE) that you can run in order to process the new commands, previously defined in the TEST.CLD command description file.

5.4.2.2 Running the User Application - To execute the newly created image file, you use the DCL command:

```
$ RUN USEREXAMP
```

This command invokes the USEREXAMP image, placing you in your newly defined application called TEST. You are now able to use your newly defined command verbs.

The following prompt will be displayed on your screen.

```
TEST>
```

This prompt (TEST>) indicates that your program is running and ready to accept input.

You can enter a command string, specifying any of the command verbs defined in the TEST.CLD file. For example, you can enter the following command string:

```
TEST> SEND
```

This command string will be interpreted as valid and will display the following text on your screen:

```
SEND COMMAND

TEST>
```

You can also include a parameter describing the file specification that you wish to include with the SEND verb. In this case, you could enter the following command string:

```
TEST> SEND MESSAGE.TXT
```

This command string will also be interpreted as valid and will display the following text on your screen:

```
SEND COMMAND

FILE-SPEC = MESSAGE.TXT

TEST>
```

If you wish to use the /EDIT qualifier (in addition to a file-spec parameter) in conjunction with the SEND command string, you could enter the following command:

```
TEST> SEND/EDIT MESSAGE.TXT
```

In this case, the following text will be displayed on your screen:

```
SEND COMMAND

/EDIT is present

FILE-SPEC = MESSAGE.TXT

TEST>
```

You may now want to enter a different command verb, defined by your application, for example, SEARCH. In this case, you enter the following command string:

```
TEST> SEARCH
```

The SEARCH command string will invoke a different routine than the one defined by the verb SEND. In this case, the following text will be displayed on your screen:

```
SEARCH COMMAND

TEST>
```

Like the SEND command, the SEARCH command permits you to include a parameter in your command string. For example, you can enter the following string:

```
TEST> SEARCH MYFILE
```

You will receive the following display on your terminal screen:

```
SEARCH COMMAND

SEARCH_STRING = MYFILE

TEST>
```

Unlike the SEND command, the SEARCH commands accepts no qualifiers. If you attempt to include a qualifier (such as /EDIT) in the SEARCH command string, CLI$DCL_PARSE will signal the following error:

```
%CLI-W-NOQUAL, qualifier not allowed on this command

TEST>
```

Finally, you can use the EXIT command to get out of your TEST utility. To do this, you simply enter the following:

```
TEST> EXIT
```

When you enter this command, you will receive the following display:

```
     $
```

The dollar sign ($) prompt indicates that you are now back in the DCL command language, which you originally used to invoke the TEST application (or utility).

Note that the EXIT command accepts no parameters and no qualifiers.

CHAPTER 6

DISK QUOTA UTILITY (DISKQUOTA)

The Disk Quota Utility (DISKQUOTA) is a system management tool available to any user maintaining a volume for controlling the usage of disk volumes. Table 6-1 summarizes the DISKQUOTA commands by format and function. See the VAX/VMS System Management and Operations Guide for usage information.

Table 6-1: DISKQUOTA Command Summary

Format	Function
ADD uic [/PERMQUOTA=quota] [/OVERDRAFT=quota-plus]	Adds an entry to the quota file
CREATE	Creates a quota file for a volume that does not currently contain one
DISABLE	Suspends quota operations on a volume
ENABLE	Resumes quota operations on a volume
EXIT	Returns the user to DCL command level
HELP	Lists the DISKQUOTA commands
MODIFY uic [/PERMQUOTA=quota] [/OVERDRAFT=quota-plus]	Changes an entry in the quota file
REBUILD	Reconstructs the usage counts for all entries
REMOVE uic	Deletes an entry from the quota file
SHOW uic	Displays quotas and usage counts
USE device	Specifies the volume to be acted upon

The sequence of commands for creating a new quota file typically includes:

1. USE -- To name the volume (can be omitted if the volume is the user's default device)

2. CREATE -- To create the quota file

3. MODIFY -- To set the quota and overdraft for UIC [0,0]

4. REBUILD -- To add entries for existing files (can be omitted on an empty volume)

5. ADD -- To add entries (one ADD command per entry) for UICs not automatically added during Step 4

The sequence of commands for modifying an existing quota file typically includes (1) a USE command followed by (2) an ADD, MODIFY, or REMOVE command for each entry being changed.

6.1 INVOKING AND TERMINATING DISKQUOTA

The following command invokes the utility:

```
$ RUN SYS$SYSTEM:DISKQUOTA
```

The system responds with the following prompt:

```
DISKQ>
```

You can then enter any of the commands listed in Table 6-1. These commands follow the standard rules of grammar as specified in the VAX/VMS Command Language User's Guide.

To terminate DISKQUOTA, you enter the DISKQUOTA command EXIT or press CTRL/Z.

6.2 DISKQUOTA COMMANDS

The following sections describe the DISKQUOTA commands in alphabetical order.

6.2.1 ADD Command

The ADD command adds an entry to the quota file. It has the following format:

```
ADD uic [/PERMQUOTA=quota]
        [/OVERDRAFT=quota-plus]
```

uic

A valid UIC.

quota

A positive integer that provides a quota for the specified UIC; defaults to the value of quota in the entry for [0,0].

quota-plus

A positive integer that provides an overdraft value for the specified UIC; defaults to the value of quota-plus in the entry for [0,0].

The usage count for a new entry is initialized to 0.

In the following example, the user sets the quota for UIC [300,211] to 200 and the overdraft to 50 (for an absolute limit of 250 blocks):

```
     DISKQ>ADD [300,211] /PERMQUOTA=200/OVERDRAFT=50
```

The ADD command requires write access to the quota file.

6.2.2 CREATE Command

The CREATE command creates a new quota file. It has the following format:

```
     CREATE
```

A quota file must not already be present on the volume being used. The CREATE command should be immediately followed by a MODIFY command for UIC [0,0] with quota and overdraft set to reasonable default values.

The CREATE command requires write access to the volume's MFD, as well as any of the following: SYSPRV privilege, a system UIC, or ownership of the volume.

6.2.3 DISABLE Command

The DISABLE command suspends the maintenance and enforcement of quotas on a volume. It has the following format:

```
     DISABLE
```

The DISABLE command requires the SYSPRV privilege, a system UIC, or ownership of the volume.

6.2.4 ENABLE Command

The ENABLE command resumes quota operations on a volume. It has the following format:

```
     ENABLE
```

The ENABLE command requires the SYSPRV privilege, a system UIC, or ownership of the volume.

6.2.5 EXIT Command

The EXIT command returns the user to DCL command level. It has the following format:

```
     EXIT
```

You can also return to DCL command level by pressing CTRL/Z.

6.2.6 HELP Command

The HELP command lists and explains the DISKQUOTA commands. It has the following format:

```
     HELP [command [/qualifier]]
```

command
: Name of a DISKQUOTA command.

/qualifier
: Name of a DISKQUOTA command qualifier.

6.2.7 MODIFY Command

The MODIFY command changes an entry in the quota file. It has the following format:

```
     MODIFY uic [/PERMQUOTA=quota]
                [/OVERDRAFT=quota-plus]
```

uic
: A UIC with an entry on the quota file.

quota
: A positive integer that specifies a quota for the specified UIC.

quota-plus
: A positive integer that specifies an overdraft for the specified UIC.

If the disk quota is less than the current usage count, a warning message is issued. (The new quota does take effect, however.)

In the following example, the user sets the permanent quota for UIC [300,211] to 300 blocks, while making no change to the overdraft:

```
     DISKQ>MODIFY [300,211] /PERMQUOTA=300
```

The MODIFY command requires write access to the quota file.

6.2.8 REBUILD Command

The REBUILD command reconstructs the usage counts for all entries on the volume. It has the following format:

```
     REBUILD
```

The REBUILD command automatically adds entries for files owned by UICs with no entries in the quota file, setting their quotas and overdrafts to the values of the defaults in UIC [0,0]. While the REBUILD command is executing, file activity on the volume is frozen -- no files can be created, deleted, extended, or truncated. For this reason, the command should be used judiciously, normally in the following situations:

- Established files -- When a quota file is created on a volume with existing files, the REBUILD command should be run before the ADD commands. REBUILD adds entries and records the existing usage for all UICs with detected usage.

- Usage not updated -- If usage counts were not updated for some period due to suppression or suspension of the DISKQUOTA feature, REBUILD should be run to correct the usage.

The REBUILD command requires write access to the quota file, as well as any of the following: the SYSPRV privilege, a system UIC, or ownership of the volume.

6.2.9 REMOVE Command

The REMOVE command deletes an entry from the quota file. It has the following format:

```
     REMOVE uic
```

uic
A UIC with an entry in the quota file.

If the usage count for the UIC is not 0, a warning message is issued. (The UIC is removed, however.) UIC [0,0] cannot be removed.

The following example deletes UIC [300,211] from the quota file:

```
     DISKQ>REMOVE [300,211]
```

The REMOVE command requires write access to the quota file.

6.2.10 SHOW Command

The SHOW command displays quotas, overdrafts, and usage counts. It has the following format:

```
     SHOW uic
```

uic
A UIC with an entry in the quota file.

An asterisk wild card character (*) can be used in specifying the UIC, as illustrated in the following examples:

Command	Description
SHOW [300,211]	Show user 211 in group 300
SHOW [300,*]	Show all users in group 300
SHOW [*,211]	Show all users with a member number of 211
SHOW [*,*]	Show all users

The SHOW command requires no privileges to show one's own quota, overdraft, and usage count, but otherwise requires read access to the quota file.

6.2.11 USE Command

The USE command specifies the volume to be acted upon. It has the following format:

```
USE device
```

device
A physical device name or a logical name equated to a physical device name.

If USE is not specified, DISKQUOTA uses the user's default device. USE can be specified more than once during a session to work with quota files on more than one volume.

Any volume in a volume set can be specified. The volume actually operated on, however, is relative volume 1.

The following examples specify the volume to be acted upon as (1) a volume on a physical device and (2) a volume on the physical device equated to a logical name:

```
1.  DISKQ>USE DMA2:

2.  DISKQ>USE X2_RESEARCH_DATA
```

6.3 ERROR MESSAGES

The *VAX/VMS System Messages and Recovery Procedures Manual* lists messages issued by the Disk Quota Utility, and provides explanations and suggested user actions.

CHAPTER 7

DISK SAVE AND COMPRESS UTILITIES (DSC)

The Disk Save and Compress (DSC) utilities are used to back up and restore disk volumes that have been formatted and initialized as Files-11 Structure Level 1 or Structure Level 2 volumes. There are three DSC utilities:

- DSC2 saves, compresses, and restores Files-11 Structure Level 2 disk volumes. DSC2 runs online under the control of the VAX/VMS operating system.

  NOTE

  DSC2 should not be used to save the running system disk.

- DSC1 performs the same functions as DSC2 for Files-11 Structure Level 1 disk volumes. DSC1 runs online under the control of the VAX/VMS operating system.

  NOTE

  DIGITAL does not intend to support the use of the DSC utilities on VAX/VMS for VAX/VMS Version 4.0 and beyond. Users should convert to the use of the Backup Utility described in Chapter 3.

This chapter describes the uses and features of the DSC utilities; it is organized into four sections:

- Section 7.1, Typical Uses of DSC, introduces the most common uses of the DSC utilities.
- Section 7.2, Specifying DSC Commands, defines the DSC command line format and the functions of the DSC file qualifiers.
- Section 7.3, Using DSC Programs, demonstrates, with examples, some of the typical uses for the DSC utilities.
- Section 7.4, Auxiliary Procedures for DSC Operations, describes procedures useful to DSC users.

- Section 7.5, DSC Messages and Error Recovery Procedures, defines all DSC-generated messages and indicates how you can recover from DSC-related error conditions.

To use the DSC utilities, you should be a VAX/VMS operator and be familiar with the following manuals:

- The software installation guide for your VAX-11 processor
- VAX/VMS System Management and Operations Guide
- VAX/VMS Command Language User's Guide

For information about Files-11 Structure Level 1 and Structure Level 2 disk volumes, refer to:

- VAX/VMS System Management and Operations Guide, which describes the Files-11 disk structures in the context of backing up public disk volumes.
- Introduction to VAX-11 Record Management Services, which describes the Files-11 disk structures in the context of general disk organization.
- VAX-11 Record Management Services Reference Manual, which explains VAX-11 RMS-related record, file, and volume concepts and formats.

7.1 TYPICAL USES FOR DSC UTILITIES

You use the DSC programs to back up and restore entire disk volumes including the system files that define the volume structure. The disk volumes can be single-device volumes or disk volume sets (which consist of one or more single-device disks bound together).

One typical use of the DSC utilities is restoring volumes that were backed up. The purpose for using a DSC program to back up a disk volume is to save a copy of the volume in case the data on the original volume is corrupted. When you back up a disk volume onto another disk, you create a usable copy of the original disk. When you back up a disk volume onto a tape set, you create a file that can later be used to restore the original disk.

Other typical uses for the DSC utilities, as described in the following sections, are:

- Backing up public or private disk volumes
- Compressing the files on a public or private disk volume
- Regulating disk bad-block information
- Comparing the contents of two volumes
- Transporting volumes

7.1.1 Backing Up the VAX/VMS System Disk

Use the Backup Utility described in Chapter 3 to back up a VAX/VMS system disk.

7.1.2 Backing Up Public or Private Disk Volumes

Use the online DSC programs DSC2 (for Files-11 Structure Level 2 volumes) or DSC1 (for Files-11 Structure Level 1 volumes) to back up public and private disk volumes.

NOTE

> It is recommended that the Backup Utility described in Chapter 3 be used to back up public and private disk volumes.

To copy the contents of public or private disk volumes to magnetic tape, refer to the examples in this chapter.

7.1.3 Compressing the Files on a Public or Private Disk Volume

To compress files on a disk volume (if backing up to tape) you back up the volume and then immediately restore it. Because of the way DSC programs execute, you cannot back up a volume without that volume being compressed. How much compression occurs depends upon how the volume has been used. For example, there may be very little compression on a volume that has been used as a private volume; but there may be a great deal of compression on a volume that has become fragmented through general use.

NOTE

> It is recommended that the Backup Utility described in Chapter 3 be used to compress files on a disk volume.

7.1.4 Regulating Disk Bad Block Information

You may need to establish and keep current the bad block information on the disk devices at your installation.

Bad block information is established for a disk when it is initialized as a Files-11 volume. Section 7.2 describes how the DCL command INITIALIZE command formats and labels a disk as a Files-11 volume; refer to the VAX/VMS Command Language User's Guide for a complete description of the INITIALIZE command.

As a disk device is used, the probability of a disk data error increases. To ensure that only good data blocks are allocated to users of a disk, you may need to establish a procedure that revises the bad block information on a disk device. One such procedure could be:

1. Run a DSC program to back up a disk volume, either to magnetic tape or to another disk (that contains up-to-date bad block information).

2. Run the Bad Block Locator Utility (BAD) program (described in Chapter 4) to determine the number and location of bad blocks

on the disk that you have just backed up. You supplement the bad block information on the disk with the bad blocks located by BAD.

3. Run the DSC program again to restore the original volume.

7.1.5 Comparing the Contents of Two Volumes

You can use a DSC program to compare the contents of two disk volumes or the contents of a disk volume and a tape set. For example, suppose you are using DSC to make ten copies of a disk volume. You could execute the DSC program ten times, each time copying from the original volume to a new volume, and each time specifying the Verify qualifier to ensure that the volumes are identical. (Refer to Section 7.3.3 for information on the use of the Verify qualifier.)

Or, you could save time by using the DSC Compare qualifier. The procedure would be:

1. Back up the original volume using DSC with the Verify qualifier.

2. Using the output volume from the previous DSC operation each time as the input volume to the next operation, execute DSC nine more times, without the Verify qualifier.

3. Finally, use DSC with the Compare qualifier to compare the contents of the first output volume and the last output volume. If the volumes compare successfully, all ten copies are good; if the volumes do not compare, the operations must be repeated.

You must decide whether the time saved by using this procedure is worth the (slight) risk of complementary errors being carried throughout the operations.

NOTE

Because the DSC programs do not support tape-to-tape transfers, they cannot be used to compare tape sets.

7.1.6 Transporting Volumes

The DSC1 program provides a way to transport the contents of Files-11 Structure Level 1 volumes between VAX/VMS installations and between VAX/VMS and RSX-11M, RSX-11M-PLUS, and IAS installations. For example, you may need to deliver the contents of several disk volumes to another system. If it is impractical to physically deliver the disk volumes, you can use the DSC1 program to save the volumes on tape sets and deliver the tape sets to the other installation. The operator at that installation can then restore the volumes from the tape sets using the DSC1 program.

Similarly, the DSC programs can be used to transport Files-11 Structure Level 2 volumes between VAX/VMS systems. However, the use of the Backup Utility (described in Chapter 3) is recommended for this operation.

7.1.7 Device Transfers Supported by DSC Programs

The DSC programs support single- and multivolume file transfers between disk devices and between disk and 9-track magnetic tape devices. DSC programs do not support transfers between magnetic tape devices. All devices supported by VAX/VMS, as listed in Appendix A, are supported by the DSC programs.

Note an important distinction between disk-to-disk transfers and disk-to-tape transfers. When a DSC program is used to back up a disk to another disk, that output disk is a usable disk volume. However, a tape set produced by a DSC program has no use, except to be input to another DSC operation`that is used to restore a disk from that tape set.

7.2 SPECIFYING DSC COMMANDS

This section describes how to invoke and terminate the DSC programs, enter DSC command strings, and define DSC operations.

7.2.1 Invoking and Terminating DSC1 and DSC2

To invoke DSC1 or DSC2, type one of the following commands in response to the DCL prompt.

For DSC1:

```
     $ RUN SYS$SYSTEM:DSC1
```

For DSC2:

```
     $ RUN SYS$SYSTEM:DSC2
```

The utility will reply with a prompt, indicating that it is ready to accept a command string. The prompts are DSC> and DSC2>. You enter a command string (followed by a RETURN) for the operation you want performed. If the operation is successful, no completion message will be displayed, and the utility will prompt for another command string.

Terminate DSC by typing CTRL/Z in response to the DSC prompt.

7.2.2 Specifying the DSC Command String

The DSC command string specifies the operation that the DSC program is to perform. The command string format is identical for the two DSC programs. Note that the left side of the equal sign in this format denotes output parameters, and the right side denotes input parameters.

```
     ddcu:  [,ddcu:...] [dsclabel] [/qualifier(s)]=

                          ddcu:  [,ddcu:...] [dsclabel] [/RW]
```

ddcu

The physical device(s) to or from which data is to be transferred, where dd is the 2-character device code, c is the device controller letter, and u is the device unit number, followed by a colon (:). For example:

```
MTA0:
```

If there is more than one output device, specify each and separate them with commas. For example:

```
MTA0:,MTA1:,MTB3:
```

dsclabel

An optional file label for the DSC file created when a disk-to-magnetic tape operation is performed. If specified, the label must be a 1- to 12-character alphanumeric label. If the label is not specified in the output specification (disk-to-tape operation), the volume label of the input disk volume becomes the output DSC file label. If the label is not specified in the input specification (tape-to-disk operation), the DSC programs begin the operation with the first file on the first input device in the command string.

For disk-to-disk operations, the label you specify becomes the volume label of the output disk. If not specified, the output disk volume label remains the same as the input disk volume label.

Specify the label before any qualifiers.

/qualifier(s)

The output file qualifiers determine the operations to be performed. They are defined in Table 7-1. They can be specified in any order; for example, /DENS=1600/VE has the same effect as /VE/DENS=1600.

/RW

The only valid input qualifier is /RW. When you specify /RW, the DSC program first rewinds the tape reel mounted on the first input device in the command string and then transfers data from it.

Table 7-1: DSC Output File Qualifiers

Format	Name and Function
/AP	The Append qualifier appends files to the tape volume specified by the first output device in the command string. Use /AP with output tapes only. /AP is required if you want to preserve the information on the output tape and the tape is either at beginning-of-tape or you have also specified the /RW output qualifier.
/BAD=MAN	The Add Bad Blocks qualifier allows you to supplement the output disk volume's bad block file with manually entered bad blocks.

(continued on next page)

Table 7-1 (Cont.): DSC Output File Qualifiers

Format	Name and Function
/BAD=NOAUTO	The Ignore Bad Block File qualifier allows you to ignore the bad block file on the output disk for this operation. Using this qualifier results in an allocated, but empty, bad block file on the output disk volume.
/BAD=MAN:NOAUTO	The Replace Bad Block File qualifier allows you to replace the output disk's bad block file with manually entered bad blocks during the DSC operation.
/CMP	The Compare qualifier does not produce data transfers. When you use /CMP, you are comparing the contents of the input device with the contents of the output device, and identifying any differences that may be present.
/DENS=1600	The Density qualifier allows you to create output tape volumes recorded at 1600 bits per inch (bpi). If not specified, output tapes are recorded at 800 bpi.
/RW	The Rewind qualifier causes the output tape volume to be rewound before data is written to it. Any data on the output tape is overwritten unless the /AP qualifier is also specified.
/VE	The Verify qualifier causes the DSC program to compare the contents of the output and input volumes and identify differences between them. The verify operation occurs after the data transfer has completed.

7.3 USING DSC PROGRAMS

This section demonstrates common uses of the DSC programs, including:

- Setting up for DSC operations
- Using DSC file labels
- Using DSC qualifiers

7.3.1 Setting Up for DSC Operations

There are several factors to consider when setting up for DSC operations:

- User privileges required. If you are not the owner of a volume that is to be backed up, you need the VOLPRO privilege to mount the volume using the MOUNT/FOREIGN command.
- Foreign mounting of output devices. All DSC output devices (both disk and tape) must be mounted with the DCL command MOUNT using the /FOREIGN qualifier.

- Valid scratch devices. Ensure that all output devices are valid scratch devices. Remember that most DSC operations destroy the original contents of the output volumes.

- Physical device names. Use only physical device names in input and output device specifications. Using logical names, although valid, increases the risk of inadvertently specifying an input volume as an output device. DSC programs initialize output disks before transferring data to them.

- Placed files. Since the DSC programs do not support files created with placement controls, do not execute DSC to save or compress disks unless you and the volume's owner agree that file placement controls can be destroyed. For information about file placement controls, refer to the RMS-11 User's Guide or the VAX-11 Record Management Services Reference Manual.

- Index file placement. When you use the INITIALIZE command to initialize a disk volume, you can specify the placement of the index file for the volume's directory structure by means of the /INDEX qualifier. (See the VAX/VMS Command Language User's Guide for information on INITIALIZE.) However, when you use DSC to initialize an output disk, it always places the index file at the beginning of a disk volume.

- Allocation errors when backing up a disk. An allocation error that occurs during a back-up of a disk usually indicates that the output disk contains too many bad blocks and does not have enough contiguous space for some large contiguous file. Retry the operation, using another output disk.

- Single disk systems. If your installation is configured with only one disk drive and no magnetic tape drive, you must use the stand-alone Backup Utility described in Chapter 3.

7.3.2 Using File Labels

For disk-to-tape operations, use DSC file labels to identify tape volumes as DSC-produced files.

If you do not specify an output file label, the DSC programs use the input disk volume label to construct DSC file label(s) for the output tape volume(s). When you specify an output DSC file label, the DSC programs construct output tape file labels from the label you provide.

If you do not specify an input file label during a restore operation from tape, the DSC programs transfer the first DSC-produced file from the first input tape device in the command string. When you specify an input file label during a restore operation from tape, the DSC programs search for that file and begin the operation, if the file is found. Otherwise, the following error message is generated:

```
DSC -- *FATAL* 75 TAPE FILE file label NOT FOUND
```

For disk-to-disk operations, you may optionally use a DSC file label to create a new volume label for the output disk.

Example: Saving a Private Volume Using DSC2

You are asked to save an RP06 volume with the disk volume label FORTVOLV3 on tape. The DSC file label for the back-up tape set is to be FORTBACK1. The back-up is expected to consume two full reels of tape.

Mount the input disk on DBA1 and the output tapes on MTA1 and MTA2. Invoke DSC2 and then enter the following command string in response to the DSC prompt:

```
DSC2>MTA1:,MTA2:FORTBACK1=DBA1:
```

This operation results in an output tape set containing the multivolume file FORTBACK1.

Had you not specified the output file label, DSC2 would have used the input disk's volume label to construct the output file label, which would then be FORTVOLV3.

Example: Restoring a Private Volume Using DSC2

To restore the volume containing the file labeled FORTBACK1 onto an RP06, mount the input tape set, FORTBACK1, on MTA1 and MTA2 and the RP06 on DBA3. Invoke DSC2 and then enter the following command string in response to the DSC prompt:

```
DSC2>DBA3:=MTA1:,MTA2:FORTBACK1
```

This operation results in a restored RP06 on DBA3 with the disk volume label FORTVOLV3. Because you specified an input file label, DSC2 searched for that file before beginning the data transfers to the output disk. Had you not specified an input file label, DSC2 would have attempted to transfer the first DSC-produced file it found on MTA1.

In this case, FORTBACK1 was the only DSC-produced file on the tape. If there were more than one DSC-produced tape file on the volume, however, you would risk copying the wrong file to the output disk by not specifying a file label.

7.3.3 Using the Verify Qualifier

Use the Verify (/VE) output qualifier when you want DSC to save, then verify the copy in the same operation. Specifying /VE directs the DSC program to compare each logical block on the input and output volumes and to identify blocks that do not compare.

Note the difference between the use of the Compare qualifier (refer to Section 7.3.7) and the Verify qualifier. DSC operations involving the Compare qualifier produce no data transfers, but an operation involving the Verify qualifier results in an output volume, then a verification of that volume.

Example: Backing Up and Verifying a Disk Volume Using DSC2

You are asked to back up an RP06 disk onto a tape, verifying the copy and using the input disk's volume label (FORTDATA1) as the output tape file label.

Mount the RP06 on DBB5 and the output scratch tape on MTA2. Invoke DSC2 and then enter the following command string in response to the DSC prompt:

```
DSC2>MTA1:/VE=DBB5:
```

The copy operation is completed before the verify operation begins. When the copy is completed, the output tape is rewound, and DSC2 issues the following message as it begins the verify operation:

```
DSC -- 45 STARTING VERIFY PASS
```

As the verify operation proceeds, warning messages are issued if any logical blocks do not compare. The messages are similar to the following example:

```
     DSC -- *WARNING* VERIFICATION ERROR ON DBB5:
     FILE ID 000623,000004,000000,VBN 000000,000001
```

Note that the output tape is not rewound at the end of the DSC operation unless more than a single volume (tape) is required. In that case, all tape volumes will be rewound and unloaded when the verification is complete.

Example: Backing Up and Verifying a Disk Volume Set Using DSC2

You are asked to back up onto magnetic tape and to verify the copy of a disk volume set consisting of two RK07 disk volumes, mounted on DMA1 and DMA2. The volume set label is MASTER_SET. You have one tape drive, MTA1, available. The back-up will consume two reels of tape.

Invoke DSC2 and enter the following command string to back up the first volume in the set:

```
     DSC2>MTA1:MASTER_SET/VE=DMA1:
```

The result of this operation is that the contents of DMA1 are copied to the tape on MTA1 until that reel is full. Then the tape is rewound, and the copy (to that point) is verified. The tape is then rewound and unloaded. DSC2 issues the following message:

```
     DSC -- 44 MOUNT REEL 2 ON MTA1:  AND HIT RETURN
```

At this point, remove the first backup tape and mount another on MTA1. When you press RETURN, DSC resumes the back-up operation and issues the following message:

```
     DSC -- 46 RESUME COPYING
```

The copy operation continues until the rest of DMA1 has been transferred. Then the verify operation resumes. DSC2 issues the message:

```
     DSC -- 45 STARTING VERIFY PASS
```

DSC2 then verifies the rest of DMA1. At the end of the operation, the tape on MTA1 is rewound and unloaded. Mount a new tape on MTA1 and enter another command string to back up the volume that is mounted on DMA2:

```
     DSC2>MTA1:MASTER_SET/VE=DMA2:
```

This second operation transfers DMA2 to magnetic tape.

7.3.4 Using the Density Qualifier

Use the Density (/DENS=1600) output qualifier when you want output tape sets to be recorded at 1600 bpi. If you do not specify the Density qualifier, output tapes are recorded at the default density of 800 bpi.

Note that the DSC programs automatically read DSC-produced tapes at the recorded density when restoring disks from tape sets.

Example: Backing Up a Disk to Tape at 1600 bpi

Using DSC2, you are to back up the system disk, an RP06 with the volume label MYVMSRL1, onto a tape set recorded at 1600 bpi. The DSC file label for the tape set is to be MONDAY; the tape drive available for the back-up is MTA2.

Mount the RP06 on device DBA0 and the scratch reel on MTA2. Invoke DSC2 and enter the following command line in response to the DSC2 prompt:

```
DSC2>MTA2:MONDAY/DENS=1600=DBA0:
```

The result of this operation is an output tape set with the DSC file label MONDAY, recorded at 1600 bpi. You would specify this file label in a DSC2 operation to restore the system disk from the tape set.

7.3.5 Using the Rewind Qualifier

Use the Rewind (/RW) qualifier as an input qualifier when you want to restore a disk from a tape set. Typically, you specify /RW when:

- You are certain that the first DSC-produced file on the first mounted input tape is the correct file.
- You have specified an input DSC file label in the command string and you want the first input tape rewound and searched for that file label before the data transfer occurs

Use the Rewind qualifier as an output qualifier when you want to save a disk onto a tape set. Typically, you would specify /RW when you want output tapes rewound to beginning-of-tape before any data transfers occur.

Example: Backing Up an RP06 Volume onto a Tape Set Using DSC1

You are asked to back up an RP06 disk that is a Files-11 Structure Level 1 volume. The RP06 is on DBA6, and two tape drives, MTA1 and MTA2, are available for the back-up. The DSC file label for the output tape set is to be MRPMON.

Invoke DSC1 and enter the following command string in response to the DSC prompt:

```
DSC>MTA1:,MTA2:MRPMON/RW=DBA6:
```

The result of this operation is a copy of the RP06 on as many tape reels as are necessary to contain the volume. The tape set created has the DSC file label MRPMON. Because you specified /RW, the first tape mounted on MTA2 is rewound before data is transferred to it. Subsequent tapes mounted on MTA2 are also rewound before data is transferred.

Example: Restoring an RP06 Volume from a Tape Set

To restore the RP06 in the example above from the tape set, mount the RP06 on device DBB2 and the first and second relative volumes of the DSC-produced file labeled MRPMON on MTA2 and MTA1. Enter the following command string:

```
DSC>DBB2:=MTA2:,MTA1:MRPMON/RW
```

The result of this operation is a restored RP06 on device DBB2. The DSC1 program checks for the correct DSC-produced file label and volume sequence number after rewinding the tape on MTA2 and starts the transfer. The restore operation continues between tapes on MTA2 and MTA1. When the contents of one tape are transferred, it is rewound and unloaded as the transfer continues from the next tape. When the last tape volume has been transferred, it is rewound and unloaded. As the copying proceeds from reel to reel, DSC checks the volume sequence number to ensure that the tapes are mounted in the correct order.

7.3.6 Using the Append Qualifier

Use the Append (/AP) output qualifier to add a DSC file to a tape volume that already contains other DSC-produced files. Typically, the Append qualifier is used to copy the contents of several small disk volumes to a tape set. Note that you use /AP only when appending separate DSC-produced files to an output tape set. Also, you can append DSC files to only the first volume in an output tape set.

Example: Backing Up Three Separate Disk Volumes Using DSC1

You need to back up onto tape three RK07 disk volumes with volume labels FORTVOL01, FORTVOL02, and FORTVOL03.

The first RK07 is mounted on the only available disk device, DMA1, and scratch tapes are mounted on devices MTA1 and MTA2.

Invoke DSC1 and start the first of three operations required by entering the following command string in response to the DSC prompt:

```
     DSC>MTA1:,MTA2:/RW=DMA1:
```

The result of this operation is a copy of FORTVOL01 to the tape on MTA1, with the output DSC file label FORTVOL01. The output tape has been rewound. At this time, physically dismount the first RK07 and replace it with the second RK07. To begin the second operation type the following:

```
     DSC>MTA1:,MTA2:/AP=DMA1:
```

The result of this operation is a transfer of the second RK07 volume to the tape on MTA1. If the transfer did not fill the reel, you can append FORTVOL03 to that reel, by following the same procedure. If necessary, the third volume can continue onto MTA2, and onto as many other tape volumes as necessary to complete the transfer.

If the transfer did fill the reel, that reel would be rewound and unloaded, and the transfer of FORTVOL02 would continue on MTA2. In this situation, you cannot append FORTVOL03 to the tape set, because an Append qualifier can be used only to append files to the first output volume in a tape set.

7.3.7 Using the Compare Qualifier

Use the Compare (/CMP) output qualifier to compare the contents of two single-volume disks or a single-volume disk and a DSC-produced tape set.

Example: Comparing a Disk to a Tape Set Using DSC2

You have been asked to use DSC2 to compare the contents of a two-volume tape set that has the DSC label PAYBACKUP with the RP06 disk volume labeled PAYMASTER.

Mount the RP06 on device DBA2 and the tape volumes on MTA2 and MTA3. Invoke DSC2 and enter the following command string in response to the DSC prompt:

```
DSC2>DBA2:/CMP=MTA2:,MTA3:PAYBACKUP/RW
```

The result of this operation is a listing of the blocks that do not compare. For each block that did not compare, a message similar to the following is issued:

```
DSC -- *WARNING* 42 VERIFICATION ERROR ON DBA2:
FILE ID 000221,000050,000000,VBN 000026,000013
```

This operation produces no data transfers: the contents of both input and output volumes are unchanged. Thus, the same results could have been obtained had you executed the operation by typing:

```
DSC2>MTA2:,MTA3:PAYBACKUP/RW/CMP=DBA2:
```

NOTE

When comparing a tape set with a disk that was not the input disk when the tape was created, it is recommended that the tape be specified as input.

7.3.8 Using Bad Block Qualifiers

On occasion you may need to use a DSC program to regulate the contents of a disk volume's bad block file before you restore that disk. On most disks, as described in Section 7.1.2, the last several blocks contain a description of the bad blocks on the disk. This description is left by the formatter or by the BAD Utility. By default, DSC constructs the volume's bad block file (BADBLK.SYS) from this data. The bad block file includes the bad blocks and the bad block description.

The DSC programs provide three output qualifiers for bad block manipulation.

Qualifier	Function
/BAD=MAN	Adds logical block numbers to the disk's bad block file
/BAD=NOAUTO	Ignores the disk's bad block information for the following DSC operation
/BAD=MAN:NOAUTO	Replaces the disk's bad block information with the logical block numbers entered during the operation

NOTE

> General procedures for establishing and maintaining the disk bad block file are described in the VAX/VMS System Management and Operations Guide. That manual defines the situations in which you use other methods for manipulating the contents of disk volume bad block files. See also the Bad Block Locator Utility (BAD), described in Chapter 4.

7.3.8.1 Using the BAD=MAN Qualifier - If you include the /BAD=MAN qualifier in the DSC command string, the utility will prompt you for the logical block numbers of the bad blocks that you want to add to the volume's bad block file.

```
     DSC>BAD=
```

You can enter the bad block numbers individually, pressing RETURN after each entry, or you can enter the bad block numbers in groups. For example:

```
     DSC>BAD=702.:3

     DSC>BAD=621.,622.

     DSC>BAD=4057.
```

The first command enters bad blocks numbers (in decimal) 702, 703, and 704; the second command enter blocks 621 and 622; the third command enters block 4057.

To terminate the list, press RETURN in response to the prompt DSC>BAD=.

The bad blocks entered manually become part of the disk's bad block file; however, they do not become part of the permanently recorded bad block description. This means that if the disk is later written onto by DSC, the bad blocks that you have previously entered will be forgotten.

7.3.8.2 Using the BAD=NOAUTO Qualifier - Disk bad block files prevent data corruption by not allowing the blocks marked as bad to be written into by user programs. However, there may be occasions which force you to ignore the bad block file on an output disk in order to complete a DSC operation. For example, it may be impossible to allocate a very large contiguous file on the output disk, due to bad blocks. On some disks, such as the RP06, it is possible to use the bad blocks and rely on ECC correction. (This technique is not advised where high data reliability is of paramount importance.)

Enter the following command string:

```
     DSC2>DBA2:/BAD=NOAUTO=DBA1:
```

7.3.8.3 Using the BAD=MAN:NOAUTO Qualifier - To replace a disk's bad block file with a new list of bad blocks, include the /BAD=MAN:NOAUTO qualifier in the DSC command string, as in the following example:

```
DSC2>DBA5:/BAD=MAN:NOAUTO=DBA6:
```

DSC2 prompts for the bad block numbers:

```
DSC>BAD=
```

You respond with the appropriate numbers, as shown in Section 7.3.8.1.

After you terminate the list by pressing RETURN in response to the DSC prompt, the bad blocks that you entered replace the bad blocks in the bad block file. Then disk-to-disk transfer begins.

7.4 AUXILIARY PROCEDURES FOR DSC OPERATIONS

Two procedures are useful to DSC users:

- Translating file identifications into file specifications
- Converting physical disk addresses to logical block numbers

7.4.1 Translating File Identifications into File Specifications

Many of the DSC messages report the file identification of a disk volume's file. While a file identification is useful to identify a unique file in a disk volume (or disk volume set), it does not tell you who is the owner of the file; what is the owner's UIC; what is the file name, file type, and version number; or what protection codes are associated with the file.

You may need to translate a file identification from its virtual block number format to a more readable form. You can do this using the MCR DUMP command.

For example, DSC-2 may issue the following message during a back-up operation:

```
DSC -- *WARNING* 41 I/O INPUT ERROR I ON DB1: FILE ID 000050,000002
```

This back-up operation continues to completion. To find out more information about the deleted file (it was not copied to DBA2), use the following procedure:

1. Reboot the VAX/VMS operating system.
2. Enter the following command string in response to the DCL prompt:

   ```
   $ MCR DMP TI:=DBA1:/FI:50:2/HD/BL:0
   ```

   The file header (FILE ID 000050,000002, given in the warning message above) is printed at the terminal.

From the record of this display, you can identify the file specification and judge whether the deletion was appropriate. If not, you may need to repeat the back-up or take some other action to restore the file.

7.4.2 Converting Disk Addresses to Logical Block Numbers

Some disk manufacturers provide a list of bad blocks with a disk device when it is shipped from the factory. Some disk diagnostic programs (including the program ESRAC in the VAX-11 Diagnostics Package) can be run to generate a list of bad blocks on a disk. Often, these lists identify bad blocks by cylinder, track, and sector number.

If you want to use a DSC program to enter these bad blocks into a disk's bad block file, you must convert these physical addresses into logical block numbers. The /BAD=MAN and the /BAD=MAN:NOAUTO qualifiers require logical block numbers.

To convert physical addresses to logical block numbers, use the following procedure:

1. If necessary, convert the cylinder, track, and sector addresses to decimal numbers.

2. Use the following formula to define the logical block number of the bad block:

```
LBN = (((cylinder number * tracks-per-cylinder)+ track number)
      * sectors-per-track) + sector number
```

For example, a diagnostic program has reported a data error on an RP06 disk at cylinder 198, track 5, and sector 8. An RP06 contains 19 tracks per cylinder and 22 sectors per track. Applying the formula produces the following result:

```
LB2 = (((198. * 19.) + 5.) * 22.) + 8.

    = (( 3762. + 5. ) * 22.) + 8.

    = (3767. * 22.) + 8.

    = 82874. +  8.

    = 82882.
```

The logical block number 82882. can now be used in a DSC command.

7.5 DSC MESSAGES AND ERROR RECOVERY PROCEDURES

This section defines the formats of messages displayed by the DSC programs, explains the meaning of each message, and indicates the user action needed (if any) to correct the situation that caused the message to be generated.

7.5.1 DSC Message Categories

There are four categories of DSC messages:

1. Information messages. These messages, displayed during DSC processing, provide the operator with information about the current DSC operation. For example:

```
DSC -- 45 STARTING VERIFY PASS
```

2. Instruction messages. These messages are displayed when operator action is required to continue the current DSC operation. The operation pauses until the operator performs the requested action, then resumes. For example:

   ```
   DSC -- 44 MOUNT REEL 2 ON MTA1:  AND HIT RETURN
   ```

3. Warning messages. These messages are displayed when a condition is detected that could cause a fatal error during subsequent DSC operations or could affect the validity of the operation. DSC processing continues as warning messages are displayed. For example:

   ```
   DSC -- *WARNING* 56 OUTPUT DISK DBA1:  IS NOT BOOTABLE
   ```

4. Fatal messages. Fatal messages terminate the current DSC operation; the program prompts for another command string. You must correct the condition that caused the message and retry the DSC operation or you must terminate the DSC program. For example:

   ```
   DSC -- *FATAL* 40 I/O ERROR F ON DBA2:
   PRIVILEGE VIOLATION
   000360

   DSC2>
   ```

7.5.2 Interpreting DSC Messages

Some DSC error messages, including those classified as I/O error messages, contain error codes, the meanings of which provide supplementary information about the error condition. Table 7-2 defines these codes.

Table 7-2: Error Codes in DSC Messages

Type of Message	Code	Meaning
General Error Messages	CODE A	Failed to read storage map header
	CODE B	Input data out of phase
	CODE C	Nondata block encountered
	CODE D	Input file out of phase
	CODE E	File attributes out of phase
	CODE F	File header out of phase
I/O Error Messages	A	Reading index file bit map
	B	Reading index file header
	C	Reading storage bit map

(continued on next page)

Table 7-2 (Cont.): Error Codes in DSC Messages

Type of Message	Code	Meaning
	D	Reading boot or home block
	E	Reading file header
	F	Input (or output) device
	G	Writing index file bit map
	H	Writing storage bit map header
	I	Reading input device
	J	Input tape labels
	K	Reading file attributes
	L	Reading file header
	M	Reading index file data
	N	Reading summary data
	O	Writing file header

7.5.3 DSC Messages

The general DSC messages are listed below with explanations and suggested user actions. Section 7.5.4 lists the DSC I/O messages. In both sections the following notations are used:

FILE ID n — File identifications are displayed as two or three numbers, for example:

```
FILE ID 000050,000002

FILE ID 000654,003456,000001
```

VBN n — VBNs are displayed as two numbers, for example:

```
VBN 000000,000001
```

aan: — The physical device

"label" — The DSC-produced file label

1 UNDEFINED ERROR

Explanation: An unidentifiable internal error was encountered.

User Action: First, retry the operation. If the error recurs, submit a Software Performance Report (SPR).

2 CONFLICTING DEV. TYPES

Explanation: An illegal combination of device types was specified.

User Action: Check for typographical errors in device abbreviations; make sure that disks and tape drives are not specified on the same side of the command string.

3 MIXED TAPE TYPES

Explanation: Two different types of tape drive were specified in the command string.

User Action: Reenter the command specifying only the magnetic tape drive.

4 ILLEGAL SWITCH

Explanation: The command string was entered with a qualifier that cannot be used.

User Action: Reenter the command with all qualifiers correctly specified.

5 FILE LABEL TOO LONG

Explanation: A file label consisting of more than 12 characters was specified.

User Action: Correct the file label, and retry the operation.

6 SYNTAX ERROR

Explanation: An error in the command string format occurred.

User Action: Check the command, and reenter the command in the correct order.

7 DUP. DEV. NAME

Explanation: The same device was specified more than once in the command string.

User Action: Reenter the command, specifying each device only once.

8 TOO MANY DEV'S

Explanation: More than eight devices were specified on one side of the command string.

User Action: Reenter the command, specifying no more than eight devices per side.

9 DEV. aan: NOT IN SYSTEM

Explanation: The specified device is not present in the configuration of the operating system being used.

User Action: Check the device identifier that was entered in the command string, and reenter the command.

10 DEV. aan: NOT FILES-11

Explanation: The specified input device is not formatted as a Files-11 device.

User Action: Check the input device to ensure it is the one desired, and reenter the command.

11 BAD BLOCK SYNTAX ERROR

Explanation: A syntax error occurred when bad block data was entered manually.

User Action: Check the command that was entered, and reenter it correctly.

12 BAD BLOCK COUNT TOO LARGE

Explanation: Too many bad blocks were manually entered in a single group.

User Action: Check the blocks being entered. If possible, enter one large group instead of several small groups.

13 BAD BLOCK CLUSTER OUT OF RANGE

Explanation: A manually entered bad block or group of bad blocks did not exist on the output disk.

User Action: Check the numbers of the blocks entered, and reenter them correctly.

14 OUTPUT TAPE aan: NOT AT BOT

Explanation: The specified continuation tape was not at load point.

User Action: Remount or reset the tape at load point, and reenter the command.

15 OUTPUT TAPE aan: FULL

Explanation: The specified tape is full; data cannot be appended to it.

User Action: Reenter the command, and change the output tape.

16 OUTPUT TAPE aan: NOT ONLY REEL IN SET

Explanation: An illegal append operation was attempted.

User Action: Reenter the command, and either omit the Append qualifier to write to the specified tape or change tapes.

17 TAPE aan: NOT ANSI FORMAT

Explanation: If aan: is an input tape, it is not in the correct format. If an output tape, an illegal Append qualifier was specified.

User Action: For input, check the tape format and change the tape, if necessary. For output, either change tapes or omit the Append qualifier from the command string.

18 OUTPUT TAPE aan: NOT DSC TAPE

Explanation: An append operation was attempted to a tape that was not created by DSC.

User Action: Reenter the command, and either omit the Append qualifier or change tapes.

19 TAPE aan: A CONTINUATION TAPE

Explanation: If aan: is an output tape, an illegal append operation was attempted. You can use the Append qualifier only on the first volume of a tape set. If aan: is an input tape, the tape was mounted out of sequence.

User Action: Reenter the command, and change the output tape, or reenter the command, and specify the input tapes in the correct order.

20 CANNOT DETERMINE DENSITY OF TAPE aan:

Explanation: Either the tape is recorded at a density that DSC cannot use or a hardware error has occurred.

User Action: Retry the operation. If the error recurs, notify the owner of the tape that it cannot be used. If it is determined later that the tape is recorded at the correct density, contact DIGITAL Field Service to report a possible hardware error.

21 FAILED TO FIND HOME BLOCK aan:

Explanation: A read error occurred during an attempt to copy from the input disk. Either the disk is bad, the home block is bad, or the disk is not in Files-11 format.

User Action: Check the disk in question, change disk drives if possible, and reenter the command.

22 FILE STRUCTURE LEVEL ON aan: NOT SUPPORTED

Explanation: The specified DSC utility program and the structure level of the specified volume did not agree.

User Action: Replace the device, and retry the operation.

23 I/O ERROR A ON aan:

Explanation: One or more messages will accompany this message, explaining why the specified file could not be read.

User Action: Retry the operation.

24 I/O ERROR B ON aan:

Explanation: One or more messages will accompany this message, indicating that an I/O error occurred and explaining why the file header on the device could not be read. The specified file was lost.

User Action: Retry the operation after correcting the cause of the error on the device.

25 CODE A aan:

Explanation: The file header for the storage bit map file could not be read.

User Action: The disk is unusable and therefore cannot be copied.

26 I/O ERROR C ON aan:

Explanation: One or more messages will accompany this message, explaining that an I/O error occurred during an attempt to read the specified file.

User Action: Retry the operation.

27 I/O ERROR D ON aan:

Explanation: A diagnostic message will accompany this message, indicating that a read error occurred during an attempt to read the name or boot block of the disk.

User Action: Retry the operation on a new drive.

28 RELATIVE VOLUME n OF SET NOT MOUNTED

Explanation: The specified tape is not on the system.

User Action: Mount the tape, and reenter the command.

29 Reserved

30 Reserved

31 I/O ERROR E ON aan: FILE ID n

Explanation: One or more messages will accompany this message, explaining that an I/O error occurred during an attempt to read the specified file header.

User Action: Retry the operation.

32 INPUT DEVICE aan: FILE ID n NOT PRESENT

Explanation: The specified file did not have a file header in the index file; the file was not copied.

User Action: This is a warning only. If desired, the operation can be retried on a different disk drive.

33 INPUT DEVICE aan: FILE ID n IS DELETED

Explanation: The specified file was found to be partially deleted on the input disk and was not copied.

User Action: This is a warning only. No action is required.

34 INPUT DEVICE aan: FILE ID n UNSUPPORTED STRUCTURE LEVEL

Explanation: The file's structure level recorded in the file header did not match the volume's structure level. This inconsistency is probably due to a garbled file header. There is no such file as n.

User Action: No user action is necessary.

35 INPUT DEVICE aan: FILE ID n FILE NUMBER CHECK

Explanation: An incorrect file header was read from disk causing the specified file to be lost.

User Action: Retry the operation.

36 INPUT DEVICE aan: FILE ID n FILE HEADER CHECKSUM ERROR

Explanation: Incorrect file header contents caused the specified file to be lost.

User Action: Retry the operation.

37 INPUT DEVICE aan: FILE ID n SEQUENCE NUMBER CHECK

Explanation: The sequence number was incorrect.

User Action: Retry the operation, and/or replace the disk.

38 INPUT DEVICE aan: FILE ID n SEGMENT NUMBER CHECK

Explanation: The linkage connecting file segments was broken; the specified file was lost.

User Action: Retry the operation.

39 DIRECTIVE ERROR - n

Explanation: An internal error occurred, usually the result of a system overload.

User Action: Retry the operation.

40 I/O ERROR F ON aan:

Explanation: One or more messages will accompany this message, indicating that the specified input or output device may subsequently cause an error.

User Action: This message is a warning only. No action is required unless another error message is displayed. If another error message is displayed, correct the cause of the error and reenter the command.

41 I/O ERROR I ON aan: FILE ID n, VBN n

Explanation: One or more messages will accompany this message, indicating that an I/O error occurred that resulted in bad data being read from the specified virtual block number on the indicated device.

User Action: This is a warning message only. The block specified should be examined to determine the extent of the error.

42 VERIFICATION ERROR ON aan: FILE ID n, VBN n

Explanation: This is a warning signifying that one block of the input and output devices did not match.

User Action: When the operation is complete, you should decide whether the mismatch requires that you retry the operation.

43 BAD DATA BLOCK ON aan: FILE ID n, VBN n

Explanation: A parity error occurred during an attempt to copy the block's contents from disk. The block specified on the output disk contains erroneous data.

User Action: When the copy operation is completed, the data contained in the specified block should be examined and corrected.

44 MOUNT REEL n ON aan: AND HIT RETURN

Explanation: This is an instruction only.

User Action: Mount the volume number requested on the specified tape drive, and press RETURN when ready.

45 STARTING VERIFY PASS

Explanation: This is simply a message informing you that the copy operation is complete and DSC is initiating the verify pass (/VE was specified).

User Action: No user action required.

46 RESUME COPYING

Explanation: This is simply a message informing you that the verify pass is complete (/VE was specified) and DSC is continuing the copy operation.

User Action: No user action required.

47 aan: IS WRITE LOCKED. INSERT WRITE RING AND HIT RETURN

Explanation: The tape on the specified tape drive cannot be written on until a write-enable ring is inserted.

User Action: Make sure the tape is the one you want, insert the write ring, and press RETURN.

48 INPUT FILE ON aan: WILL BE RESYNCHRONIZED

Explanation: The tape position was lost during an attempt to read the input tape. The file specified in the message, as well as some subsequent files, may be lost. Additional errors will probably occur.

User Action: Retry the operation from the beginning.

49 OUTPUT DEVICE aan: FULL FILE ID n

Explanation: The specified device is full and cannot accommodate the data following the specified file. This may mean that more data than anticipated was transferred due to an inconsistency in the input tapes. Or, the output device may contain too many bad blocks to allocate a large contiguous file.

User Action: Reenter the command, using a larger output disk.

50 OUTPUT FILE HEADER FULL ON aan: - FILE ID n

Explanation: Too many blocks on the output disk have caused inconsistencies in file header data. The specified file was lost.

User Action: Retry the operation with a different output disk.

51 OUTPUT FILE HEADER ON aan: NOT MAPPED - FILE ID n

Explanation: Space for the specified file header was not allocated. The file was lost.

User Action: Retry the operation; a new disk may be required.

52 I/O ERROR G ON aan:

Explanation: One or more messages will accompany this message, indicating that an I/O error occurred during an attempt to write the specified file.

User Action: Retry the operation.

53 FAILED TO READ FILE EXTENSION HEADER ON aan: - FILE ID n

Explanation: During an attempt to copy data from the input disk, an extension header was searched for, but not found. The remainder of the specified file was lost. A problem may exist with the input disk, or a previous I/O error may have caused an inconsistency.

User Action: Retry the operation.

54 FAILED TO ALLOCATE HOME BLOCK aan:

Explanation: The home block could not be created on the specified disk device because it has too many bad blocks.

User Action: Replace the device, and reenter the command.

55 INDEX FILE ALLOCATION FAILURE aan:

Explanation: Too many bad blocks exist to allow the allocation for the specified file.

User Action: Replace the disk, and reenter the command.

56 OUTPUT DISK aan: IS NOT BOOTABLE

Explanation: Logical block number 0 of the specified disk or tape is bad.

User Action: This is a warning only. No action is required.

57 INVALID BAD BLOCK DATA aan:

Explanation: The bad block data on the output disk is invalid.

User Action: Run the BAD Utility on the disk, manually enter bad block data, or reenter the command using a new disk.

58 BAD BLOCK FILE FULL aan:

Explanation: Too many bad blocks exist on the output disk.

User Action: Replace the disk, and reenter the command.

59 NO BAD BLOCK DATA FOUND aan:

Explanation: No bad block data exists for the specified output disk.

User Action: If bad block data is not desired, ignore the message. Otherwise, run the BAD program on the disk, manually enter bad block data, or reenter the command using a new disk.

60 OUTPUT DEVICE aan: IS A DIAGNOSTIC PACK. DO NOT USE IT!

Explanation: The specified output disk is a diagnostic pack and cannot be used.

User Action: Mount another output disk, and reenter the command.

61 CODE B ON aan: FILE ID n - VBN n EXPECTED, m FOUND

Explanation: The tape position was lost during an attempt to read the virtual block number specified. Some data may be lost.

User Action: Determine the extent of the error. If necessary, try the tape on another drive or create another tape.

62 CODE C ON aan: FILE ID n - VBN n

Explanation: The position of the tape was lost during an attempt to read the specified data file. Data beyond the virtual block number specified was lost.

User Action: Re-create the tape or retry the operation on a different tape drive.

63 CODE D ON aan: FILE ID n EXPECTED, m FOUND

Explanation: The tape position was lost during an attempt to read the specified tape. All of "n" and some of "m" were lost.

User Action: Retry the entire operation.

64 FAILED TO MAP OUTPUT FILE ON aan: FILE ID n, VBN n

Explanation: An inconsistency occurred during an attempt to write the specified file to the output disk. The file header did not specify the correct number of virtual blocks required to write the file, and the file was lost.

User Action: Retry the operation.

65 OUTPUT DISK aan: IS TOO SMALL - n BLOCKS NEEDED

Explanation: The output disk is not large enough to accommodate the data to be transferred.

User Action: Retry the operation specifying a larger output disk.

66 I/O ERROR CODE C ON aan:

Explanation: One or more messages will accompany this message, explaining that an I/O error occurred during an attempt to read the specified file.

User Action: Retry the operation.

67 I/O ERROR CODE H ON aan:

Explanation: One or more messages will accompany this message, explaining that an I/O error occurred during an attempt to write the specified file.

User Action: Retry the operation.

68 I/O ERROR CODE J ON aan:

Explanation: One or more messages will accompany this message, explaining that an I/O error occurred during an attempt to read the tape labels on the specified device.

User Action: Retry the operation on a different tape drive.

69 INPUT TAPE ON aan: MUST BE AT BOT

Explanation: The specified tape must be at the beginning of the tape (BOT) or at its load point. This message is also displayed during a verify operation to indicate that the current volume is rewinding to enable the verify pass.

User Action: If /VE was not specified, check the tape and remount at load point.

70 WRONG INPUT TAPE ON aan: EXPECTING "label", FOUND "label"

Explanation: The input tapes were specified out of sequence.

User Action: Check the tapes and reenter them in the correct order after receiving mount instructions.

71 CODE E ON aan: AFTER FILE ID n

Explanation: This is the result of a read error from tape. During an attempt to read an attribute block, some other block was accessed. The file following the file specified in the error message was lost.

User Action: Retry the operation.

72 I/O ERROR K ON aan: AFTER FILE ID n

Explanation: One or more messages will accompany this message, indicating that an I/O error occurred during an attempt to read the specified file.

User Action: Retry the operation.

73 I/O ERROR L ON aan: AFTER FILE ID n

Explanation: One or more messages will accompany this message, indicating that an I/O error occurred during an attempt to read the file header.

User Action: Retry the operation.

74 INPUT TAPE aan: RESYNCHRONIZED AT FILE ID n

Explanation: The tape position was recovered. Some data preceding the file specified was lost.

User Action: This message is usually displayed with one or more error messages, all indicating that the input tape was either read incorrectly or recorded badly. The tape should be re-created and the operation reinitiated.

75 TAPE FILE "label" NOT FOUND aan:

Explanation: The input tape specified does not contain the file identified as "label."

User Action: Check the file label and the tape, and reenter the command when the correct tape and file label are specified.

76 EXPECTED EXTENSION HEADER NOT PRESENT ON aan: - FILE ID n

Explanation: A tape read error occurred, causing the specified file to be lost.

User Action: If the error message was preceded by one or more I/O warning messages, the operation should be retried. If not, the input tape is bad and should be regenerated.

77 CODE F ON aan: AFTER FILE ID n

Explanation: This is the result of a read error from tape. During an attempt to read a file header, some other block type was accessed. The file following the file specified in the error message was lost.

User Action: Retry the operation.

78 I/O ERROR M ON aan:

Explanation: One or more messages will accompany this message, explaining why the specified file could not be read.

User Action: Retry the operation.

79 INDEX FILE DATA NOT PRESENT aan:

Explanation: During an attempt to read the input tape specified, a file other than the index file was accessed due to a tape error or an I/O error.

User Action: Re-create the tape or retry the same tape on a different tape drive.

80 I/O ERROR N ON aan:

Explanation: One or more messages will accompany this message, indicating that an I/O error occurred during an attempt to restore the index and storage map files from the specified input tape.

User Action: Retry the operation using a different input tape drive.

81 VOLUME SUMMARY DATA NOT PRESENT aan:

Explanation: Either the input tape is not a DSC tape or it contains incomplete data.

User Action: Check the tape, and reenter the command.

82 I/O ERROR O ON aan: FILE ID n

Explanation: One or more messages will accompany this message, indicating that an I/O error occurred during an attempt to write the specified file header.

User Action: Retry the operation.

83 UNSUPPORTED DSC TAPE FORMAT ON aan:

Explanation: This tape cannot be processed with this version of the DSC program.

User Action: Retry the operation. If the same failure recurs, contact DIGITAL Software Support, or submit a Software Performance Report (SPR).

7.5.4 DSC I/O Error Messages

The DSC I/O error messages are listed below.

BAD BLOCK NUMBER

Explanation: The block does not exist on the disk, an internal DSC error occurred, or the block is bad.

User Action: Retry the operation with a new disk and/or disk drive.

BAD BLOCK ON DEVICE

Explanation: A device malfunction occurred or a tape with bad data on it was used, resulting in a block containing incorrect information.

User Action: Retry the operation.

BLOCK CHECK OR CRC ERROR

Explanation: A parity error occurred, indicating that bad data may have been transferred.

User Action: Retry the operation.

DATA OVERRUN

Explanation: The physical tape used was larger than expected or got out of position, or was in the wrong format.

User Action: Make sure the tape is the right one and retry the operation.

DEVICE NOT READY

Explanation: The device was not ready or not up to speed, or a blank tape was used as an input tape.

User Action: Retry the operation after checking that the device is online and correctly mounted.

DEVICE OFF-LINE

Explanation: The device is not in the system.

User Action: Check both the device and the device specification in the command string, and reenter the command.

DEVICE WRITE LOCKED

Explanation: The disk drive is write locked.

User Action: Write enable the disk drive, and reenter the command.

END OF FILE DETECTED

Explanation: The tape position was lost.

User Action: Retry the operation.

END OF TAPE DETECTED

Explanation: The tape position was lost.

User Action: Retry the operation.

END OF VOLUME DETECTED

Explanation: The tape position was lost.

User Action: Retry the operation.

FATAL HARDWARE ERROR

Explanation: A hardware malfunction occurred.

User Action: Retry the operation; if the error recurs call DIGITAL Field Service.

ILLEGAL FUNCTION

Explanation: An operation was attempted, but DSC cannot determine what it was.

User Action: Retry the operation. If the same failure recurs, contact DIGITAL Software Support or submit a Software Performance Report (SPR).

INSUFFICIENT POOL SPACE

Explanation: The operating system is overloaded.

User Action: Retry the operation.

PARITY ERROR ON DEVICE

Explanation: A device malfunction or media incompatibility occurred.

User Action: Retry the operation.

PRIVILEGE VIOLATION

Explanation: A device has been mounted as Files-11.

User Action: Dismount the disk, mount it as a foreign volume, and retry the operation.

UNKNOWN SYSTEM ERROR

Explanation: An undefinable I/O error occurred.

User Action: Retry the operation.

CHAPTER 8

FILE TRANSFER UTILITY (FLX)

The File Transfer Utility (FLX) is a utility program that transfers files from one volume to another. FLX can be used on DOS-11, RT-11, and Files-11 formatted volumes. It converts the format of the files, as appropriate, when transferring files between volumes with different formats. For example, when transferring DOS-11 files to Files-11 volumes, FLX converts the DOS-11 files to Files-11 format.

FLX performs file transfers and format conversions from:

- DOS-11 to DOS-11 volumes
- Files-11 to Files-11 volumes
- RT-11 to RT-11 volumes
- DOS-11 to Files-11 volumes
- Files-11 to DOS-11 volumes
- Files-11 to RT-11 volumes
- RT-11 to Files-11 volumes

FLX cannot transfer files directly between DOS-11 and RT-11 volumes.

In addition to transferring files, FLX allows you to:

- Initialize DOS-11 or RT-11 volumes
- List directories of DOS-11 or RT-11 volumes
- Delete files from RT-11 volumes

FLX recognizes all Files-11 volumes on VAX/VMS devices. It recognizes DOS-11 formatted volumes on the following devices:

TE16, TU45, TU77, or TU78 magnetic tape

FLX recognizes RT-11 formatted volumes on the following devices:

- TU58 DECtape II data cartridge
- RL02 cartridge disk
- RK06 or RK07 cartridge disk
- RX01 or RX02 floppy disk

Files-11 Structure Level 1 or Files-11 Structure Level 2 volumes are the default volumes initialized by the DIGITAL Command Language (DCL) command INITIALIZE. DOS-11 and RT-11 volumes are initialized using FLX commands. Since the formats of these volumes are not recognized by VAX/VMS, the volumes must be mounted foreign, that is, by use of the /FOREIGN qualifier. See the VAX/VMS Command Language User's Guide for more information on the INITIALIZE and MOUNT commands.

You can use FLX interactively or through a command procedure. FLX allows only one level of indirect command file specification.

8.1 INVOKING AND TERMINATING FLX

To invoke FLX, enter the following in response to the DCL prompt:

```
$ RUN SYS$SYSTEM:FLX
```

The utility responds with the prompt:

```
FLX>
```

You can now enter any FLX command string. To return to DCL at any time, type CTRL/Z.

8.2 FLX COMMAND STRING

Formats for specifying FLX functions vary, as described in Sections 8.4 through 8.6. The following are possible formats for a FLX command; each element in the command string is explained below.

```
device-spec/qualifier
device-spec=file-spec/qualifier
file-spec/qualifier
/qualifier
```

device-spec

The device name and directory for the FLX output device. It takes the form:

```
ddu:[directory]
```

The device name (dd) can be any of the 2-character device codes listed below.

DOS-11

Device Code	Device
MT	TE16, TU45, TU77
MS	TS-11
MF	TU78

RT-11

Device Code	Device
DD	TU58
DL	RL02
DM	RK06, RK07
DY	RX02
CS	RX01 (VAX-11/780 console floppy)

Files-11

Device codes for Files-11 devices are listed in Appendix A.

The u is the unit number of the device. FLX does not recognize alphabetic controller designators. You must convert them to RSX-11M unit numbers when specifying devices to FLX; for example, MTA1 must be specified as MT1. The controller designator can still be used, however, in ALLOCATE and MOUNT commands referring to volumes to be used with FLX.[1]

The colon (:) acts as the device name terminator and must follow the device code.

The directory field is optional. The directory specification is subject to restrictions depending on the medium to which it applies.

RT-11 volumes accept no directory specification at all.

DOS-11 volumes accept only directories in the user identification code (UIC) format, for example [310,22]. The two numbers are octal, and must be in the range 0 through 377. When the directory is specified in an input file specification, either number or both may be indicated by a wild card character. If you do not specify a directory, FLX uses your current default directory, if it is in UIC format; otherwise, it uses your process's UIC.

Files-11 volumes accept the standard form of VAX/VMS directory specification documented in the VAX/VMS Command Language User's Guide. Wild card characters may be specified only with a single level of directory or with the UIC format. If you do not specify a directory, FLX uses your current default directory.

file-spec

The file specification for an input or output file.

Wild card characters are valid only for input file specifications. Version numbers are valid only for Files-11 files and cannot be specified as wild card characters. FLX does not accept logical names in file specifications.

FLX does not permit output file specifications. The output files take the names of the input files.

RT-11 volumes use 6-character file names, plus 3-character file types; file names are truncated to six characters when files are copied into RT-11 volumes.

/qualifier

Any of the qualifiers described in Section 8.3. Multiple qualifiers can be used; their order is not important.

[1] For information on converting VAX/VMS native mode unit numbers to compatibility mode unit numbers, see the explanation of mapping physical device names in the VAX-11/RSX-11M User's Guide.

8.3 FLX QUALIFIERS

FLX uses three types of qualifiers:

- Volume format qualifiers, which specify the format of the volume on which files are stored: Files-11, DOS-11, or RT-11 volumes.
- Transfer mode qualifiers, which specify the format of a file on a non-Files-11 volume. Files can be in formatted ASCII, formatted binary, or file image format.
- Control qualifiers, which provide control functions for use in FLX operations. These qualifiers can be used to specify such items as the number of blocks to be allocated to an output file or the density of a magnetic tape.

These three types of qualifiers are described in detail in the next three sections.

8.3.1 Volume Format Qualifiers

The three volume format qualifiers are used in command strings to define the format of volumes. They can also be used by themselves after the FLX> prompt to change the default for input and output volumes. Table 8-1 describes these qualifiers.

Table 8-1: FLX Volume Format Qualifiers

Qualifier	Function
/DO	Identifies the volume as a DOS-11 formatted volume
/RS	Identifies the volume as a Files-11 formatted volume
/RT	Identifies the volume as an RT-11 formatted volume

Initially, input volumes default to DOS-11 volumes and output volumes default to Files-11 format. FLX assumes these default volume formats if you do not specify a format qualifier in the file transfer command string.

You can change the default by entering /DO or /RS on a command line by itself. To specify that the default transfer is from DOS-11 to Files-11, type:

```
FLX>/DO
```

To specify that the default transfer is from Files-11 to DOS-11, type:

```
FLX>/RS
```

If /RT is specified on one side of a command string, the default qualifier for the other side is /RS. For example:

```
FLX>DM0:/RT=DD0:SYS1.MAC
```

The input is defaulted to /RS. In the next example, the output is defaulted to /RS:

```
     FLX>DM0:=DM0:SYS1.MAC/RT
```

You cannot transfer files directly between RT-11 and DOS-11 volumes using FLX.

8.3.2 Transfer Mode Qualifiers

Transfer mode qualifiers are used to specify the format that an output file should have after it is copied or converted. FLX has three transfer mode qualifiers, one for each type of file format: formatted ASCII, formatted binary, and file image format. Format conversions can be in either direction between DOS-11 files and Files-11 files or between RT-11 files and Files-11 files. Table 8-2 describes the transfer mode qualifiers.

Table 8-2: FLX Transfer Mode Qualifiers

Qualifier[1]	Function
/FA:n	Formatted ASCII Formatted ASCII files consist of ASCII data records terminated by carriage return/line feed (CR-LF), form feed (FF), or vertical tab (VT) characters. In transfers from DOS-11 or RT-11 files to Files-11 files, CR-LF pairs are removed from the end of records. In transfers from Files-11 files to DOS-11 or RT-11 files, CR-LF pairs are added to the end of each record that does not already end with LF or FF. All null (NUL), delete (DEL), and vertical tab (VT) characters are removed from input records in any of these transfers. If you specify /FA:n with Files-11 output, fixed-length records of size n are generated. Output records are padded with null characters as necessary. If you do not specify /FA:n with Files-11 output, FLX generates variable-length records. The output record size equals the input record size. ASCII data is transferred as 7-bit codes. Bit 8 (the parity bit) of each byte is masked before transfer.
/FB:n	Formatted Binary If you specify /FB with DOS-11 or RT-11 output files, formatted binary headers and checksums are added to the records.

[1] Note that n, which specifies record size, is useful only for Files-11 volumes. If you enter it when specifying other volumes, it will be ignored. All n values are interpreted in octal unless followed by a period.

(continued on next page)

Table 8-2 (Cont.): FLX Transfer Mode Qualifiers

Qualifier[1]	Function
	Specifying /FB:n with Files-11 output produces fixed-length records of size n, up to 512 decimal bytes long. FLX pads records with null characters to reach the specified length. If you do not specify n for Files-11 output, FLX generates variable-length records. The output record size equals the input record size.
/IM:n	Image Mode Specifying an image mode transfer always produces fixed-length records. You can use the value n to indicate the desired number of bytes in the record (up to 512) for Files-11 output. If you do not specify n, FLX assumes a record length of 512 bytes.

[1] Note that n, which specifies record size, is useful only for Files-11 volumes. If you enter it when specifying other volumes, it will be ignored. All n values are interpreted in octal unless followed by a period.

FLX assumes the following default transfer modes for these file types:

Qualifier	File Type
/IM:n (Image Mode)	TSK, OLB, MLB, SYS, SML, ULB, EXE
/FB (Formatted Binary)	OBJ, STB, BIN, LDA
/FA (Formatted ASCII)	All others

8.3.3 Control Qualifiers

FLX provides control qualifiers to control file processing. Table 8-3 describes these qualifiers.

Table 8-3: FLX Control Qualifiers

Qualifier	Function
/BL:n	Indicates the number of contiguous blocks (n) to be allocated to the output file. If you do not specify /BL, the input file size is used as the output file size. /BL:n is used with RT-11 output to circumvent the normal RT-11 file allocation scheme, which allocates the largest available space on the

(continued on next page)

Table 8-3 (Cont.): FLX Control Qualifiers

Qualifier	Function
	volume for a new file. Using /BL:n with the /RT switch for the output file causes n blocks to be allocated for the output file instead of the largest available space. When FLX has finished transferring the file to the RT-11 volume and the file is closed, the output file will have the same number of blocks as the input file, less than or equal to n. If the input file size is larger than n, an error will occur. The /BL:n qualifier is normally used with the /CO qualifier, as described below. Because all RT-11 files are contiguous, the /CO qualifier need not accompany the /BL:n qualifier for RT-11 output.
/CO	Indicates that the output file is to be contiguous. The /CO qualifier is used only with disk output. When the input file is in DOS-11 format, use /BL:n with /CO (see the description of /BL:n above). When the input is a Files-11 volume or an RT-11 disk, FLX assumes /CO in transferring file types TSK, SYS, and OLB to Files-11 volumes.
/DE	With /RT, deletes files from a disk with RT-11 formatted volumes. When you specify /DE, the FLX command string needs no output specification.
/DI or /LI	Causes a directory listing to be listed on a specified output file. Use /RT with /DI or /LI to generate a directory listing of RT-11 volumes. If you do not specify an output device, the directory will be sent to SYS$OUTPUT. If you do not specify file name and file type on the input file specifications, *.* is assumed. You cannot list Files-11 volume directories using FLX.
/DNS:n	Specifies the magnetic tape density in bits per inch (bpi); n is either 800 or 1600. If n is any other number or is not specified at all, an error will occur. If you do not specify /DNS:n, the magnetic tape density will default to 800 bpi. If you specify /DNS with any device but magnetic tape, FLX will ignore the qualifier.
/FC	With FORTRAN files on Files-11-formatted output files, indicates that FORTRAN carriage control conventions should be used, that is, that FORTRAN should interpret certain characters as carriage

(continued on next page)

Table 8-3 (Cont.): FLX Control Qualifiers

Qualifier	Function
	control characters. (See the VAX-11 FORTRAN Language Reference Manual for more information on FORTRAN carriage control conventions, or see the VAX-11 Record Management Services Reference Manual for a discussion of the file access block and record attributes, which include setting carriage control.)
/ID	Requests that the current version number of FLX be printed. You can specify /ID as part of an output or input specification, or type it in response to the FLX prompt (FLX>).
/LI	Same as /DI, explained above.
/NU:n	With the /ZE and /RT qualifiers, specifies the number of directory blocks (n) to be allocated when FLX initializes an RT-11 disk. If you do not specify /NU:n, four directory blocks will be allocated. The maximum number of blocks that can be allocated is 31.
/RW and /-RW	/RW rewinds the magnetic tape before FLX begins the file transfer. /-RW causes FLX to begin the transfer without first rewinding the magnetic tape. The default is /RW. If you specify /RW or /-RW with any device other than magnetic tape, or with the qualifiers /DI, /LI, or /ZE, FLX will ignore the rewind qualifier.
/SP	Indicates that the converted file is to be spooled. /SP is used only with Files-11 output files.
/UI	Indicates that the output file is to have the same directory as the input file. Do not use /UI when you are specifying an explicit output UIC. /UI is valid only with output files in DOS-11 or Files-11 format.
/ZE	Initializes DOS-11 or RT-11 volumes. To initialize RT-11 volumes, you must also specify /RT and /NU:n. Initializing erases any files already on the device.

8.4 TRANSFERRING FILES WITH FLX

To transfer files from one volume to another, enter a command string of the form

```
     device-spec[/qualifier]=file-spec[/qualifier]...file-spec[/qualifier]
```

The FLX transfer specifies the output device on the left of the equal sign and the files to be transferred on the right of the equal sign.

In constructing the transfer command string, keep in mind the restrictions upon the various FLX qualifiers listed in Section 8.3, as well as the restrictions upon the format of device and file specifications.

8.5 DOS-11 VOLUME DIRECTORY MANIPULATION

You can display DOS-11 directory listings and initialize DOS-11 volumes using FLX qualifiers as described in the following sections.

Remember that DOS-11 volumes must be mounted foreign before you can manipulate them using FLX.

8.5.1 Displaying DOS-11 Directory Listings

The /DI qualifier described in Table 8-3 sends the directory of the DOS-11 volume specified in the input specification to the Files-11 file specified in the output specification. If you do not enter an output specification, FLX sends the directory to SYS$OUTPUT. For example:

```
     FLX>MT0:/DO/DI
```

This command lists on your terminal all files from the DOS-11 volume on the magnetic tape drive MT0.

Figure 8-1 shows a sample directory listing of a DOS-11 volume, followed by notes keyed to the figure.

```
DIRECTORY ❶      MT0:[360,27]❷
29-JUN-82 ❸
                                             ❻
FLXCHA.DOC ❹                 35.❺    29-JUN-82
PRACTW.DOC                   2.      29-JUN-82
WATSP.DOC                    3.      29-JUN-82

TOTAL OF 40.  BLOCKS IN 3.  FILES ❼
```

Figure 8-1: DOS-11 Directory Listing

Notes to Figure 8-1:

❶ The listing identifier.

❷ The device name, unit number, and UFD.

❸ The date the directory was listed.

❹ The file name and file type.

❺ The number of blocks in the file.

❻ The file creation date.

❼ The total number of blocks allocated to all files on the volume.

8.5.2 Initializing DOS-11 Volumes

You can initialize DOS-11 volumes using the /ZE qualifier. This qualifier requires only the device specification for the volume you are initializing. For example:

```
     FLX>MT0:/DO/ZE
```

This command initializes the magnetic tape on MT0 in DOS-11 format.

8.6 RT-11 VOLUME DIRECTORY MANIPULATION

You can display RT-11 directory listings, delete RT-11 files, and initialize RT-11 volumes using FLX qualifiers as described in the following sections.

Remember that VAX/VMS RT-11 volumes must be mounted foreign before you can manipulate them using FLX.

8.6.1 Displaying RT-11 Directory Listings

The /DI qualifier, when combined with the /RT qualifier, sends the directory of the RT-11 volume specified in the input specification to the Files-11 file specified in the output specification. If you do not enter an output specification, FLX will send the directory to SYS$OUTPUT. For example:

```
     FLX>DM0:*.MAC/DI/RT
```

This command lists on your terminal all files with the file type of MAC from the RT-11 volume on DM0.

Figure 8-2 shows a sample directory listing of an RT-11 volume, followed by notes keyed to the figure.

```
DIRECTORY❶      DM0❷
22-JUN-82
                                  ❻
SIPBOO.MAC❹     49.❺        22-JUN-82
< UNUSED >       6.
SIP   .MAC      10.         22-JUN-82
SIPCD .MAC       7.         22-JUN-82
< UNUSED >      21.
SIPQIO.MAC       7.         22-JUN-82
< UNUSED >   26998.

 27025.   FREE BLOCKS❼
                                  ❽
TOTAL OF 73.  BLOCKS IN 4.  FILES
```

Figure 8-2 RT-11 Directory Listing

Notes to Figure 8-2:

❶ The listing identifier.

❷ The device name and unit number.

❸ The date the directory was listed.

❹ The file name and file type; < UNUSED > indicates free space.

❺ The number of blocks in the file or free space.

❻ The file creation date; blank for free space.

❼ The total number of free blocks on the volume.

❽ The total number of blocks allocated to all files on the volume.

8.6.2 Initializing RT-11 Volumes

You can initialize RT-11 volumes using the /ZE qualifier with the /RT qualifier. The /ZE qualifier requires only the device specification for the volume you are initializing. For example:

```
FLX>DM1:/ZE/RT
```

This command initializes the RT-11 formatted volume on DM1.

When you initialize RT-11 volumes, /ZE takes an optional argument in the form:

```
/ZE:n
```

The value n specifies the number of extra words per directory entry, in addition to the 7-word default length. This capacity for increasing the length of directory entries is useful for some RT-11 applications. Note that when you increase the number of words per directory entry by specifying /ZE:n, you are reducing the number of directory entries.

Using the /NU:n qualifier with /ZE and /RT specifies the number of directory segments, n, to be allocated to the RT-11 volume. Four directory segments (consisting of two disk blocks each) are allocated by default. The maximum number of segments that can be allocated is 31(decimal). For example:

```
FLX>DM0:/ZE:2/NU:6/RT
```

This command

- Initializes the disk on DM0
- Allocates two extra words per directory entry
- Allocates six directory segments

8.6.3 Deleting RT-11 Files

You can delete files from RT-11 disks using the /DE qualifier with the /RT qualifier. The command string on which you specify /DE/RT requires the device and file specifications for the file you are deleting. For example:

```
FLX>DM1:SYS1.MAC/DE/RT
```

This command deletes the file SYS1.MAC from the RT-11 volume on DM1.

8.7 FLX MESSAGES

Errors encountered by FLX during processing are reported on the initiating terminal. The FLX messages, their explanations, and suggested user actions are listed in the VAX/VMS System Messages and Recovery Procedures Manual.

CHAPTER 9

INSTALL UTILITY (INSTALL)

The Install Utility (INSTALL) is a system management tool that installs and maintains known images. See the VAX/VMS System Management and Operations Guide for additional information on the Install Utility. Table 9-1 summarizes the INSTALL commands.

Table 9-1: INSTALL Command Summary

Format	Function
file-spec [/OPEN] [/HEADER_RESIDENT] [/PRIVILEGED[=(priv-name[,...])]] [/PROTECTED] [/SHARED] [/WRITEABLE]	Installs an image as a known image with qualifiers as specified
file-spec /DELETE	Deletes an image as a known image
file-spec /REPLACE	Associates a known image with the latest version of the image file
[file-spec] /LIST	Displays a single-line description of one specific known image or, by default, of all known images
[file-spec] /FULL	Displays a multiline description of one specific known image or, by default, of all known images
/GLOBAL	Displays all global sections
/EXIT	Terminates the session
/HELP	Describes (interactively) how to use INSTALL

9.1 INVOKING AND TERMINATING INSTALL

The following command invokes the utility:

```
     $ RUN SYS$SYSTEM:INSTALL
```

You can also run INSTALL as a foreign command. Simply provide the following definition:

```
$ INSTALL :== $INSTALL
```

Then, to invoke INSTALL, you can use the following command:

```
$ INSTALL
```

The system responds with the following prompt:

```
INSTALL>
```

You can then enter any of the commands listed in Table 9-1. These commands follow the standard rules of grammar as specified in the VAX/VMS Command Language User's Guide.

The file specification must name an existing executable or shareable image. Defaults are applied as follows:

- Device and directory -- SYS$SYSTEM
- File type -- EXE

A version number must not be specified. The highest existing version is always used. Specification of a version number produces unpredictable results (because file lookups for known images ignore version numbers).

You terminate INSTALL and return to DCL command level with the /EXIT command or by pressing CTRL/Z.

Use of any of the Install Utility functions (except /HELP and /EXIT) requires the CMKRNL privilege. The Install Utility is installed with the PRMGBL and SYSGBL privileges, which are required for the use of the /SHARED qualifier.

You can obtain help while executing INSTALL by typing /HELP.

9.2 INSTALL COMMANDS

The following sections describe the INSTALL commands according to the functions they perform.

9.2.1 Installing Known Images

You install a known image with the following command:

```
file-spec [/OPEN]
          [/HEADER_RESIDENT]
          [/PRIVILEGED[=(priv-name[,...])]]
          [/PROTECTED]
          [/SHARED]
          [/WRITEABLE]
```

file-spec
    File specification of the image being installed.

/OPEN
    Installs a permanently open known image.

/HEADER_RESIDENT
Installs a known image with a permanently resident header (native mode images only). Then, if the image is not located on the system volume, the image is made permanently open even if /OPEN is not specified.

/PRIVILEGED[=(priv-name[,...])]
Installs a known image with privileges (executable images only). The image is made permanently open even if /OPEN is not specified. Privilege names are specified as shown in Section 9.2.5. If no privilege names are specified, all privileges are assigned. Parentheses can be omitted for specification of one privilege.

/PROTECTED
Installs a known image with protected code.

/SHARED
Installs a shared known image. Causes creation of global sections for the image. The image is made permanently open even if /OPEN is not specified.

/WRITEABLE
Installs a writeable known image. If /SHARED is not also specified, this qualifier is ignored.

The following example installs a permanently open, shared known image:

```
INSTALL> DBA1:[MAIN]STATSHR /OPEN/SHARED
```

The next example installs a permanently open known image with privileges:

```
INSTALL> GRPCOMM /OPEN/PRIVILEGES=(GROUP,GRPNAM)
```

Any process running GRPCOMM receives the GROUP and GRPNAM privileges for the duration of the execution of GRPCOMM. The full name of GRPCOMM is assumed to be SYS$SYSTEM:GRPCOMM.EXE.

9.2.2 Deleting Known Images

You delete a known image with the following command:

```
file-spec /DELETE
```

file-spec
File specification of an image installed as a known image.

The image's entry on the known file list and any global sections created for the image are deleted. The image itself (that is, the image file) remains unaffected. Writeable non-CRF global sections are written back to disk upon their removal as known images.

The following example deletes a known image:

```
INSTALL> DBA1:[MAIN]STATSHR /DELETE
```

9.2.3 Replacing Known Images

You replace a known image with the following command:

```
file-spec /REPLACE
```

file-spec
File specification of an image installed as a known image.

The image's entry on the known file list becomes associated with the latest version of the image file.

The replace function cannot be used to assign additional attributes to a known image. If an image is installed without a resident header, for example, a specification of /REPLACE/HEADER_RESIDENT does not add this attribute. You must delete the known image and reinstall it with the desired attributes.

The following example replaces a known image:

```
     INSTALL> GRPCOMM /REPLACE
```

The full name of the file specification is assumed to be SYS$SYSTEM:GRPCOMM.EXE.

9.2.4 Listing Known Images

You display information on known images and global sections with the following commands:

- [file-spec] /LIST
- [file-spec] /FULL
- /GLOBAL

file-spec
File specification of an image installed as a known image.

/LIST
Displays a one-line description of the specified known image or (if no file specification is made) all known images; see Figure 9-1.

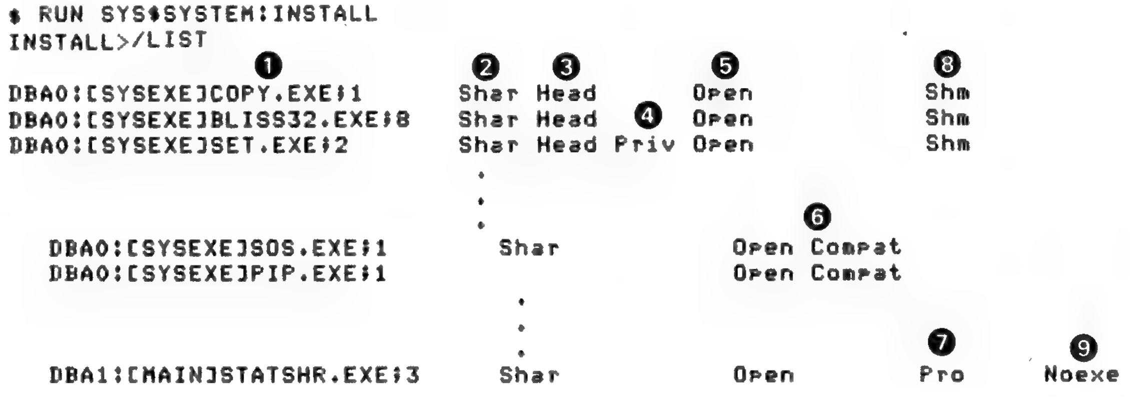
/FULL
For display of a multiline description of the specified known image or (if no file specification is made) all known images; see Figure 9-2.

/GLOBAL
For display of all global sections (whether or not the section was created as a result of INSTALL); see Figure 9-3.

9.2.5 Specifying Privileges

The currently available privileges are as follows:

CMKRNL
CMEXEC
SYSNAM
GRPNAM
ALLSPOOL
DETACH
DIAGNOSE
LOG_IO
GROUP
ACNT
PRMCEB
PRMMBX
PSWAPM
ALTPRI
SETPRV
TMPMBX
WORLD
MOUNT
OPER
EXQUOTA
NETMBX
VOLPRO
PHY_IO
BUGCHK
PRMGBL
SYSGBL
PFNMAP
SHMEM
SYSPRV
BYPASS
SYSLCK



```
$ RUN SYS$SYSTEM:INSTALL
INSTALL>/LIST
                 ❶             ❷    ❸         ❺             ❽
DBA0:[SYSEXE]COPY.EXE;1      Shar Head      Open           Shm
DBA0:[SYSEXE]BLISS32.EXE;8   Shar Head    ❹ Open           Shm
DBA0:[SYSEXE]SET.EXE;2       Shar Head Priv Open           Shm
                               .
                               .
                               .                   ❻
   DBA0:[SYSEXE]SOS.EXE;1        Shar          Open Compat
   DBA0:[SYSEXE]PIP.EXE;1                      Open Compat
                                 .
                                 .
                                 .                        ❼       ❾
   DBA1:[MAIN]STATSHR.EXE;3      Shar          Open       Pro     Noexe
```

Figure 9-1: Known Image Display -- Brief Description

Notes on Figure 9-1:

❶ File specification of known image

❷ Known image is shared

❸ Known image has permanently resident header

❹ Known image has privileges

❺ Known image is permanently open

❻ Known image runs in compatibility mode

❼ Known image contains protected code

❽ Known image is installed in multiport memory

❾ Known image is not executable -- that is, it is a shareable image

```
$ RUN SYS$SYSTEM:INSTALL
INSTALL>USERS /FULL
DBA0:[SYSEXE]USERS.EXE;1          Head Priv Open ❶
    Access Count      = 30 ❷
    Privileges        = GROUP WORLD ❸
    Entry adr/size    = 8006EBF0/64 ❹
    Window adr/size   = 800A17A0/48 ❺
    Header adr/size   = 8006A8FC/368 ❻
```

Figure 9-2: Known Image Display -- Full Description

Notes on Figure 9-2:

❶ Same as for brief description (see Figure 9-1)

❷ Number of times known file entry has been accessed

❸ Translation of the privilege mask; appears only if image installed with privileges

❹ Address (in hexadecimal) of image's entry on known file list and the size of the entry in bytes

❺ Address (in hexadecimal) of image's directory window and the size of the window in bytes; appears only if image installed permanently open

❻ Address (in hexadecimal) of image's resident header and the size of the header in bytes; appears only if image installed with permanently open resident header

❼ Not shown in example - if image runs in compatibility mode, second line reads: Compatibility type = 0000

```
$ RUN SYS$SYSTEM:INSTALL
INSTALL>/GLOBAL
                                          ❶
                                  System Global Sections
      ❷                ❸         ❹   ❺          ❻   ❼              ❽
 LBRSHR_004      (01000001) WRT CRF        PRM SYS      Pagcnt/Refcnt=1/0
 CRFSHR_003      (010003E8) WRT CRF        PRM SYS      Pagcnt/Refcnt=1/1
                                .
                                .
                                .         ❾
 27 Global Sections Used, 828 Global Pages Used, 3268 Global Pages Unused
                                              ❿
                       Global Sections in Multiport Memory 'SHM'
                                                                   ⓫
 VMSRTL_001       (010003FC)                PRM SYS     Creator Port=0
           Basepfn/Pagcnt=00000000/7 ⓬
           Port 0 PTE Refcnt=35 ⓭
                                .
                                .
                                .
  659 Global Pages Used, 1329 Global Pages Unused
  15 Global Sections Used, 17 Global Sections Unused ⓮
```

Figure 9-3: Global Sections Display

Notes on Figure 9-3:

❶ Display of global sections in local memory

❷ Name of global section

❸ Version number (in hexadecimal) of global section; high-order byte (01 in CRFSHR_003) contains major identification and low-order bytes (0003E8 in CRFSHR_003) contain minor identification

❹ The global section is writeable

❺ The global section is copy-on-reference

❻ The global section is permanent; TMP indicates a temporary global section

❼ The global section is system-wide; GRP and a group number indicate a group-wide section

❽ Number of pages in the section and number of times pages from the section are currently referenced. Note that the total is not merely the sum of all global sections' page counts. This is because each section requires one global page per section page, plus two additional global pages, with the total per global section rounded to an even number

❾ Number of global sections created, number of global pages used, and number of global pages unused in local memory

❿ Display of global sections in shared multiport memory unit; one such display appears for each attached multiport memory unit; the multiport memory unit 2 in the example is named SHM

⓫ Number of the port from which the global section was created

⓬ Base page frame number (in hexadecimal) and number of pages in the section

⓭ Page table entry (PTE) reference count; one appears for each port where the reference count is not 0

⓮ Number of global pages used, number of global pages unused, number of global sections used, and number of global sections unused in shared memory

9.3 ERROR MESSAGES

The VAX/VMS System Messages and Recovery Procedures Manual lists messages issued by the Install Utility, and provides explanations and suggested user actions.

CHAPTER 10

LIBRARIAN UTILITY (LIBRARY)

The Librarian Utility allows you easy access to libraries. Libraries are indexed files that contain frequently used modules of code or text. There are four different types of libraries -- object, macro, help, and text. The library type indicates the type of module that the library contains. Section 10.1 describes the four types of libraries. Each library contains indexes that store information about the library's contents, including the type, location, and modification history of the individual modules. Section 10.1.2 describes library indexes.

The Librarian consists of two parts: (1) the DIGITAL Command Language (DCL) command LIBRARY (see Section 10.2), which you use to replace and maintain modules in an existing library, or to create a new library; and (2) a collection of Librarian routines (see Section 10.5) that you can call from a program to initialize and open a library, and to retrieve, insert, and delete modules.

10.1 LIBRARIES

The following sections describe the four library types -- object, macro, help, and text -- and the contents of library indexes.

10.1.1 Types of Libraries

There are four types of libraries, distinguished by their file types:

- Object libraries (file type OLB) contain frequently called routines and are used as input to the linker. The linker searches the object module library whenever it encounters a reference it cannot resolve from the specified input files. See the VAX-11 Linker Reference Manual for more information on how the linker uses libraries. Users should note that an image library does not contain the shareable image.

  Object libraries can also be identified in a way other than by file type. For example, you can create the object library LIB.xxx by typing:

  ```
  $ LIB/CREAT/OBJ LIB.xxx *.obj
  ```

  and then access it by typing:

  ```
  $ LIB/LIST LIB.xxx
  ```

- Macro libraries (file type MLB) contain macro definitions used as input to the assembler. The assembler searches the macro library whenever it encounters a macro that is not defined in the input file. See the VAX-11 MACRO Language Reference Manual for information on defining macros.

- Help libraries (file type HLB) contain help modules, that is, modules that provide user information about a program. You can retrieve help messages in your program by calling the appropriate Librarian routines. See Section 10.3 for information about creating help modules for insertion into help libraries.

- Text libraries (file type TLB) contain any sequential record files that you want to retrieve as data for your program. Your programs can retrieve text from text libraries by calling the appropriate Librarian routines. See the /INSERT qualifier in Section 10.2.2 for information about inserting text files into text libraries.

- Shareable image libraries (file type OLB) contain the symbol tables of shareable images which are used as input to the VAX/VMS Linker utility. See Section 10.2.2 for a description of the /SHARE qualifier and Section 10.4 for information on how to create a shareable image library.

You use DCL commands to manipulate libraries in their entirety; for example, the DELETE, COPY, and RENAME commands delete, copy, and rename libraries, respectively. For more information on file maintenance, see the VAX/VMS Command Language User's Guide.

Section 10.5.14 describes the treatment of character case for the different types of libraries.

10.1.2 Structure of Library Indexes

Every library contains a library header that describes the contents of the library. The information in the library header includes:

- The type of library
- The number of indexes and their location in the file
- The version number of the Librarian
- The library's creation date and time
- The date and time of the last update
- The library's preallocated size and its current size

Each module has a module header that contains information about the module, including its type, its attributes, and its date of insertion into the library.

Libraries can contain more than one index. All libraries contain an index for the module name table (MNT). Object module libraries also contain an index called a global symbol table (GST) that is a list of the global symbols defined in each of the library modules.

The MNT catalogs modules by module name, rather than by the name of the input file that contained the inserted module. The only exception to this procedure occurs with text libraries, for which the file name of the input file containing the text automatically becomes the module name. See the description of the /MODULE qualifier in Section 10.2.1.

10.2 THE DCL LIBRARY COMMAND

This section describes how to create, modify, and maintain libraries using the DCL command LIBRARY. This information also appears in the VAX/VMS Command Language User's Guide.

The purpose of the LIBRARY command is to maintain object, macro, help, or text libraries. The command's default operation is to replace modules. By specifying various qualifiers, you can also use the LIBRARY command to create and modify libraries, and to insert, delete, extract, and list library modules and symbols.

10.2.1 LIBRARY Command String

To use the LIBRARY command, enter the following command string in response to the DCL prompt:

```
LIBRARY/qualifier(s)  library-file-spec  [input-file-spec[/MODULE=
module-name][,...]]
```

/qualifier(s)

The function(s) to be performed by the LIBRARY command. Section 10.2.2 describes the qualifiers in detail.

library-file-spec

The name of the library you want to create or modify. This parameter is required. If you do not specify a library file, you will be prompted for one as follows:

```
$_Library:
```

No wild card characters are allowed in the library file specification.

If the file specification or a qualifier in the command line does not include a file type, the LIBRARY command assumes a default type of OLB, indicating an object library.

NOTE

Any attempt to modify a library that was created by the VAX-11 Version 1.0 Librarian results in an automatic compression into the new format introduced with Version 2.0. The compression occurs before the requested modification. (See the /COMPRESS qualifier in Section 10.2.2.) Furthermore, libraries created before Version 2.0 that have not been modified or compressed will appear in a different format when listed by the /LIST qualifier.

input-file-spec[,...]

The names of one or more files that contain modules you want to insert into the specified library.

Whenever you include an input file specification, the LIBRARY command either replaces or inserts the modules contained in the input file(s) into the specified library. The input file specification is required when you specify either /REPLACE (the LIBRARY command's default operation) or /INSERT, which is an optional qualifier. If you do not specify an input file when you use these qualifiers, you will be prompted for it as follows:

```
$_File:
```

When you use the /CREATE qualifier to create a new library, the input file specification is optional. If you include an input file specification with the /CREATE qualifier, the LIBRARY command first creates a new library, and then inserts the contents of the input file(s) into the library.

Note that the /EXTRACT qualifier does not accept an input file specification.

If you specify more than one input file, separate the file specifications with commas (,). The LIBRARY command will then insert the contents of each file into the specified library.

If any file specification does not include a file type and if the command string does not indicate one, the LIBRARY command will assume a default file type of OBJ, designating an object library. You can control the default file type by specifying the appropriate qualifier as indicated below.

Qualifier	**Default File Type**
/HELP	HLP
/MACRO	MAR
/OBJECT	OBJ
/TEXT	TXT
/SHARE	EXE

Note also that the file type you specify with the library file specification affects the default file type of the input file specification, provided that you do not specify the /CREATE qualifier. For example, if the library file type is HLB, MLB, OLB, or TLB, the input file type default will be HLP, MAR, OBJ, or TXT, respectively.

Wild card characters are allowed in the input file specification(s).

/MODULE=module-name

The module-name of a text module you want to insert or replace. When you are inserting text modules into a library, the input file that you specify is taken to be a single module. Therefore, the file name of the input file specification becomes the module-name. If you want the file you are inserting to have a module-name different from the input file name, use the /MODULE qualifier to name the added module.

You can also use the /MODULE qualifier to enter a text module interactively. If you specify the logical name SYS$INPUT as the input file, and issue the /MODULE qualifier, the LIBRARY command will insert the text you enter from the terminal into the specified library module. To terminate the terminal input, press CTRL/Z.

Remember that the /MODULE qualifier is an input file qualifier; it assumes that you are either replacing or inserting a new text module. Therefore, the qualifiers that remove modules /EXTRACT, /DELETE, /REMOVE -- are incompatible with /MODULE.

10.2.2 Command Qualifiers

When using the LIBRARY command, you can specify qualifiers that request more than one function in a single command, with some restrictions. Generally, you cannot specify multiple qualifiers that request incompatible functions. The qualifiers that perform library functions, related qualifiers, and qualifier incompatibilities are summarized in Table 10-1.

Table 10-1: LIBRARY Command Qualifier Compatibilities

Qualifier	Related Qualifiers	Incompatible Qualifiers
/COMPRESS	/OUTPUT	/CREATE, /EXTRACT
/CREATE[1]	/SQUEEZE,[2] /GLOBALS[3] /SELECTIVE_SEARCH[3]	/COMPRESS, /EXTRACT
/CROSS_REFERENCE	/ONLY	/EXTRACT, /LIST
/DELETE	---	/EXTRACT
/EXTRACT	/OUTPUT	/COMPRESS, /CREATE /DELETE, /INSERT /LIST, /REMOVE /REPLACE
/INSERT	/SQUEEZE,[2] /GLOBALS[3] /SELECTIVE_SEARCH[3]	/EXTRACT
/LIST	/FULL, /NAMES,[3] /ONLY /HISTORY, /BEFORE /SINCE	/EXTRACT /CROSS_REFERENCE
/REMOVE[3]	---	/EXTRACT
/REPLACE	/SQUEEZE,[2] /GLOBALS[3] /SELECTIVE_SEARCH[3]	/EXTRACT
/MODULE[4]	/TEXT	/EXTRACT, /DELETE /REMOVE

1. The /CREATE, /INSERT, and /REPLACE qualifiers are compatible; however, if you specify more than one, /CREATE takes precedence over /INSERT, and /INSERT takes precedence over /REPLACE. The related qualifiers for /CREATE are applicable only if you enter one or more input files.

2. This qualifier applies only to macro libraries.

3. This qualifier applies only to object libraries.

4. This file qualifier applies only to text libraries.

The qualifiers listed in Table 10-1 are described in detail below.

/BEFORE[=time]

Specifies that only those modules dated earlier than a particular time be listed. You can specify an absolute time, but you must observe the rules for absolute time values. Type HELP SPECIFY ABSOLUTE_TIME for details on specifying absolute times.

This qualifier is used in conjunction with the /LIST qualifier. If you omit the /BEFORE qualifier, you obtain all the modules regardless of date. However, if you specify /BEFORE without a date or time, the default provides the files created prior to today.

/COMPRESS[=(option[,...])]

Requests the LIBRARY command to perform either of the following functions:

- Recover unused space in the library resulting from module deletion
- Reformat a library created by the VAX/VMS Version 1.0 or Version 2.0 Librarian into the Version 2.0 or Version 3.0 format

When you specify /COMPRESS, the LIBRARY command by default creates a new library with a version number one higher than the existing library. Use the /OUTPUT qualifier to specify an alternate name for the compressed library.

Specify one or more of the following options to alter the size or format of the library, overriding the values specified when the library was created:

BLOCKS:n   Specifies the number of 512-byte blocks to be allocated for the library

GLOBALS:n   Specifies the maximum number of global symbols the library can contain (for object module libraries only)

HISTORY:n   Specifies the maximum number of library update history records that the library will maintain

KEEP   Copies library update history records and any additional user data in the module header, to the compressed library

KEYSIZE:n   Changes the maximum length of module names or global symbol names

MODULES:n   Specifies the maximum number of modules or macros the library can contain

VERSION:n   Changes the library format for the library by specifying 2 for VAX/VMS 2.0 library format or 3 for VAX/VMS 3.0 library format

If you specify more than one option, separate them with commas and enclose the list in parentheses.

/CREATE[=(option[,...])]

Requests the LIBRARY command to create a new library. When you specify /CREATE, you can optionally specify a file or a list of files that contain modules to be placed in the library.

By default, the LIBRARY command creates an object module library; specify /MACRO, /HELP, or /TEXT to indicate that the library is a macro, help, or text library.

Specify one or more of the following options to control the size of the library, overriding the system defaults:

BLOCKS:n Specifies the number of 512-byte blocks to be allocated for the library. By default, the LIBRARY command allocates 100 blocks for a new library.

GLOBALS:n Specifies the maximum number of global symbols the library can contain initially. By default, the LIBRARY command sets a maximum of 128 global symbols for an object module library. (Macro, help, and text libraries do not have a global symbol directory; therefore, the maximum for these libraries defaults to 0.)

HISTORY:n Specifies the maximum number of library update history records that the library will maintain.

KEYSIZE:n Specifies the maximum name length of modules or global symbols. By default, the LIBRARY command limits the names of global symbols and object, macro, and text modules to 31 characters. The name length limit for help modules is 15 characters. When you specify a key-size value, remember that VAX-11 MACRO and the VAX-11 Linker do not accept module names or global symbol names in excess of 31 characters.

MODULES:n Specifies the maximum number of modules the library can contain. By default, the LIBRARY command sets an initial maximum of 512 modules for an object module library and 256 modules for a macro, help, or text library.

VERSION:n Creates a library with the desired format by specifying 2 for VAX/VMS 2.0 library format or 3 for VAX/VMS 3.0 library format.

If you specify more than one option, separate them with commas and enclose the list in parentheses.

/CROSS_REFERENCE[=(option[,...])]

Requests a cross-reference listing of an object library.

If you omit this qualifier, cross-reference listings will not be provided. However, if you specify /CROSS_REFERENCE without specifying an option, you will obtain cross-reference listings by default that contain only symbols by name and symbols by value.

You can specify one or more of the following options:

ALL	Specifies that all types of cross-references are required
MODULE	Specifies a cross-reference listing of both the global symbol references in the module and the global symbol definitions
NONE	Specifies that no cross-reference listing is required
SYMBOL	Provides a cross-reference listing by symbol name
VALUE	Provides a cross-reference listing of symbols by value

If you specify more than one option, separate the options with commas and enclose the list in parentheses.

/DELETE=(module[,...])

Requests the LIBRARY command to delete (physically remove) one or more modules from a library. You must specify the names of the modules to be deleted. If you specify more than one module, separate the module names with commas and enclose the list in parentheses.

Wild card characters are allowed in the module specification.

If you specify the /LOG qualifier with /DELETE, the LIBRARY command will issue the message:

```
%LIBRAR-S-DELETED, MODULE module-name DELETED FROM library-name
```

/EXTRACT=(module[,...])

Copies one or more modules from an existing library into a new file. If you specify more than one module, separate the module names with commas and enclose the list in parentheses.

Wild card characters are allowed in the module specification.

If you specify the /OUTPUT qualifier with /EXTRACT, the LIBRARY command will write the output into the file specified by the /OUTPUT qualifier. If you specify /EXTRACT and do not specify /OUTPUT, the LIBRARY command will write the file into a file that has the same file name as the library and a file type of OBJ, MAR, HLP, or TXT, depending on the type of library.

/FULL

Requests a full description of each module in the module name table. Use this qualifier with the /LIST qualifier to request a list of each library module in the format:

```
module-name [Ident  nn]dd   Inserted dd-mmm-yyyy   hh:mm:ss   [n symbols]
```

The identification number and the number of symbols appear only in object libraries.

If an update history is maintained for the library, then /LIST/FULL/HISTORY will list the module name in the update history records.

/GLOBALS
/NOGLOBALS

Controls, for object module libraries, whether the names of global symbols in modules being inserted in the library are included in the global symbol table.

By default, the LIBRARY command places all global symbol names in the global symbol table. Use /NOGLOBALS when you do not want the global symbol names in the global symbol table.

/HELP

Indicates that the library is a help library. When you use the /HELP qualifier, the library file type defaults to HLB and the input file type defaults to HLP.

/HISTORY

When used with /LIST, requests that record headers of the library update history records be listed in the format:

```
username operation[1] n modules on dd-mmm-yyy hh:mm:ss
```

When used with /LIST/FULL, requests that the update history headers and the names of updated modules be listed.

/INSERT

Requests the LIBRARY command to add the contents of one or more files to an existing library. If an object module input file consists of concatenated object modules, the LIBRARY command will create a separate entry for each object module in the file; each module name table entry reflects an individual module name. If a macro or help file specified as input contains more than one definition, the LIBRARY command will create a separate entry for each one, naming the module name table entries according to the names specified on the .MACRO directives or in the key-1 name in the HELP format (see Section 10.3.1).

In text libraries, unlike object, macro, and help libraries, the input file contains data records of undefined contents. Therefore, the Librarian catalogs the entire input file as a single module using the file name (not the directory or file type) as the module name. If you want to rename the inserted module, use the /MODULE qualifier described in Section 10.2.1.

When the LIBRARY command inserts modules into an existing library, it checks the module name table before inserting each module. If a module name or global symbol name already exists in the library, an error message will be issued and the module or symbol will not be added to the library.

To insert or replace a module in a library regardless of whether a current entry exists with the same name, use the /REPLACE qualifier (the default operation).

[1] Operation can be replaced, inserted, or deleted

/LIST[=file-spec]
/NOLIST

Controls whether the LIBRARY command creates a listing of the contents of the library.

By default, no listing is produced. If you specify /LIST without a file specification, the LIBRARY command will write the output file to the current SYS$OUTPUT device. If you include a file specification that does not have a file type, the LIBRARY command will use the default file type of LIS.

If you specify /LIST with qualifiers that perform additional operations on the library, the LIBRARY command will create the listing after completing all other requests; thus, the listing reflects the status of the library after all changes have been made.

When you specify /LIST, the LIBRARY command provides, by default, the following information about the library:

```
Directory of OBJECT library _DBB0:[LIBRAR]LIBRAR.OLB;1 on 14-JUN-1982 10:08:28
Creation date:  12-JUN-1982 19:40:36       Creator:  VAX-11 Librarian V03-00
Revision date:  14-JUN-1982 16:04:58       Library format:  3.0
Number of modules:      15                 Max. key length:  31
Other entries:          73                 Preallocated index blocks:     35
Recoverable deleted blocks:     15         Total index blocks used:       12
Max. update history records:    10         Update history records:        5
```

/LOG
/NOLOG

Controls whether the LIBRARY command verifies each library operation. If you specify /LOG, the LIBRARY command will display the module name, followed by the library operation performed, followed by the library file specification. Examples of the /LOG qualifier appear in the descriptions of /DELETE and /REPLACE.

/MACRO

Indicates that the library is a macro library. When you specify /MACRO, the library file type defaults to MLB and the input file type defaults to MAR.

/NAMES
/NONAMES

Controls, when /LIST is specified for an object module library, whether the LIBRARY command lists the names of all global symbols in the global symbol table as well as the module names in the module name table.

The default is /NONAMES, which does not list the global symbol names. If you specify /NAMES, each module entry name will be displayed in the format:

```
module    "module-name"

global-symbol      global-symbol      global-symbol      global-symbol
      .                  .                  .                  .
      .                  .                  .                  .
      .                  .                  .                  .
```

If the library is a macro, help, or text library and you specify /NAMES, no symbol names will be displayed.

/OBJECT

Indicates that the library is an object module library. This is the default condition. The LIBRARY command assumes a library file type of OLB and an input file type of OBJ.

/ONLY=(module[,...])

Specifies the individual modules on which the LIBRARY command can operate. When you use the /ONLY qualifier, the LIBRARY command lists or cross-references only those modules specified.

If you specify more than one module, separate the module names with commas and enclose the list in parentheses.

Wild card characters are allowed in the module name specification.

/OUTPUT=file-spec

Specifies, when the /EXTRACT, /COMPRESS, or /CROSS_REFERENCE qualifiers are specified, the file specification of the output file.

For /EXTRACT, the output file contains the modules extracted from a library; for /COMPRESS, the output file contains the compressed library; for /CROSS_REFERENCE, the output file contains the cross-reference listing.

No wild card characters are allowed in the file specification.

If you omit the file type in the file specification, a default will be used depending on the library function qualifier and, in some cases, the library type qualifier as shown below.

Qualifier	Library Type Qualifier	Default File Type
/COMPRESS	/HELP	HLB
	/MACRO	MLB
	/OBJECT	OLB
	/TEXT	TLB
/CROSS_REFERENCE	---	LIS
/EXTRACT	/HELP	HLP
	/MACRO	MAR
	/OBJECT	OBJ
	/TEXT	TXT

/REMOVE=(symbol[,...])

Requests the LIBRARY command to delete one or more entries from the global symbol table in an object library. If you specify more than one symbol, separate the symbols with commas and enclose the list in parentheses.

Wild card characters are allowed in the symbol specification.

To display the names of the deleted global symbols, you must also specify the /LOG qualifier.

/REPLACE

Requests the LIBRARY command to replace one or more existing library modules with the modules specified in the input file(s). The LIBRARY command first deletes any existing library modules with the same name as the modules in the input file. Then, the new version of the module is inserted in the library. If any modules contained in the input file do not have a corresponding module in the library, the LIBRARY command will insert the new modules in the library.

This is the LIBRARY command's default operation. If you specify an input file parameter, the LIBRARY command will either replace or insert the contents of the input file into the library. If you use the /LOG qualifier with the /REPLACE qualifier, the LIBRARY command will display, in the following form, the names of each module that it replaces or inserts:

```
%LIBRAR-S-REPLACED, MODULE module-name REPLACED IN library-file-spec


%LIBRAR-S-INSERTED, MODULE module-name INSERTED IN library-file-spec
```

/SELECTIVE_SEARCH

Defines the input files being inserted into a library as candidates for selective searches by the linker. If you specify /SELECTIVE_SEARCH, the modules will be selectively searched by the linker when the library is specified as a linker input file; only the global symbol(s) in the module(s) referenced by other modules are included in the symbol table of the output image file.

/SHARE

Indicates that the library is a shareable image library. The LIBRARY command assumes a file type of OLB for the shareable image symbol table library and EXE for the input files. See Section 10.4 for additional information.

/SINCE[=time]

Specifies that only those modules dated later than a particular time be printed. You can specify an absolute time, but you must observe the rules for absolute time values. Type HELP SPECIFY ABSOLUTE_TIME for details on specifying absolute times.

This qualifier is used in conjunction with the /LIST qualifier. If you omit the /SINCE qualifier, you obtain all the modules regardless of date. However, if you specify /SINCE without a date or time, the default provides the files created since today began.

/SQUEEZE
/NOSQUEEZE

Control whether the LIBRARY command compresses individual macros before adding them to a macro library. When you specify /SQUEEZE, which is the default, trailing blanks, trailing tabs, and comments are deleted from each macro before insertion in the library.

Use /SQUEEZE with the /CREATE, /INSERT, and /REPLACE qualifiers to conserve space in a macro library. If you want to retain the full macro, specify /NOSQUEEZE.

/TEXT

Indicates that the library is a text library. When you use the /TEXT qualifier, the library file type defaults to TLB and the input file type defaults to TXT.

/WIDTH=n

Controls the screen display width (in characters) for listing global symbol names. Specify the /WIDTH qualifier with the /NAMES qualifier to limit the line length of the /NAMES display.

The default display width is the width of the listing device. The maximum width is 132.

Examples

1.
```
$ LIBRARY/CREATE TESTLIB ERRMSG,STARTUP
```

The LIBRARY command creates an object module library named TESTLIB.OLB and places the modules ERRMSG.OBJ and STARTUP.OBJ in the library.

2.
```
$ LIBRARY/INSERT TESTLIB SCANLINE
$ LINK TERMTEST TESTLIB/LIBRARY
```

The LIBRARY command adds the module SCANLINE.OBJ to the library TESTLIB.OLB. The library is specified as input to the linker by using the /LIBRARY qualifier on the LINK command. If the module TERMTEST.OBJ refers to any routines or global symbols not defined in TERMTEST, the linker will search the global symbol table of library TESTLIB.OLB to resolve the symbols.

3.
```
$ LIBRARY/EXTRACT=(ALLOCATE,APPEND)/OUTPUT=MYHELP SYS$HELP:
  HELPLIB.HLB
```

The LIBRARY command specifies that the modules ALLOCATE and APPEND be extracted from the help library HELPLIB.HLB and output to the file MYHELP.HLP.

4.
```
$ LIBRARY/CROSS_REFERENCE=ALL/OUTPUT=SYS$OUTPUT LIBRAR
```

The LIBRARY command requests a cross-reference listing of the object library LIBRAR.OLB. The cross-reference listing is displayed at the terminal. The listing includes cross-references by symbol, by value, and by module.

5.
```
$ LIBRARY/REMOVE=(LIB_EXTRCT_MODS,LIB_INPUT_MAC)/LOG LIBRAR
```

The LIBRARY command requests the removal of the global symbols LIB_EXTRCT_MODS and LIB_INPUT_MAC from the object library LIBRAR.OLB. The /LOG qualifier requests that the removal of the symbols be confirmed by messages.

6.
```
$ LIBRARY/MACRO/CREATE=(BLOCKS:40,MODULES:100) MYMAC TEMP
$ MACRO MYMAC/LIBRARY,CYGNUS/OBJECT
```

The LIBRARY command creates a macro library named MYMAC.MLB from the macros in the file TEMP.MAR. The new library has room for 100 modules in a 40-block file. If the input file contains multiple macros, each macro will be entered in the new library.

The MACRO command assembles the source file CYGNUS.MAR; the /LIBRARY qualifier specifies the library MYMAC.MLB as an input file. If the source file CYGNUS contains any macro calls not defined within the file, the assembler will search the library.

7. `$ LIBRARY/LIST=MYMAC.LIS/FULL MYMAC.MLB`

   The LIBRARY command requests a full listing of the macro library MYMAC; the output is written to a file named MYMAC.LIS.

8. `$ LIBRARY/INSERT/TEXT TSTRING SYS$INPUT/MODULE=TEXT1`

   The LIBRARY command inserts a module named TEXT1 into the text library TSTRING.TLB. The input is taken from SYS$INPUT.

9. `$ LIBRARY/LIST/NAMES/ONLY=$ONE/WIDTH=80 SYMBOLIB`

   The LIBRARY command requests a full listing of the module $ONE, contained in the object library SYMBOLIB.OLB. The /WIDTH qualifier requests that the global-symbol display be limited to 80 characters per line.

10.3 HELP LIBRARIES

Help messages are a convenient means of providing specific information about a program to an interactive user. The help messages are stored as modules in help libraries. Your programs can access the help modules by calling the appropriate Librarian routines described in Section 10.5. In this way, users of your program can quickly retrieve relevant information about using your program.

You create help libraries in the same manner that you create object, macro, and text libraries, using the LIBRARY/CREATE command described in Section 10.2.2. However, before you can insert modules into a help library, you must format the input file so that the Librarian can catalog its individual modules. This section describes how to create input files containing help modules.

10.3.1 Creating Help Files

The input file that you insert into a help library is a text file that you build with a text editor. Each input file may contain one or more help modules. A help module is a group of help messages that relates to the same topic, or key.

Each module within a help library contains a group of related key names, or topics, numbered key-1 through key-n. The key-1 name identifies the main topic of help information; for example, the name of a command in your program that requires explanation. The key-2 through key-n names identify subtopics that are related to the key-1 name; for example, the command's parameters and/or qualifiers. This organization enables users of your program to find a general message describing how to use the command, and then optionally to select subtopics that provide additional information about the command's parameters and qualifiers.

Index keywords are in the format in which they were entered, that is, individual uppercase and lowercase characters are entered that way. A second, identically spelled keyword (but of a different or mixed character case) to the same library replaces the previous keyword. However, character case is ignored for match operations, for example, "help Sample" and "help SAMPLE" will access the same file.

10.3.2 Formatting Help Files

Each key-1 line in the module consists of the key number (1) in the first column, followed by the name of the key. Subsequent subkey lines, key-2 through key-n, consist of the subkey number followed by the name of the subkey. For example, a help module for a command might have the following two key lines:

```
1 Command name
          .
          .
          .
   help message text
          .
          .
          .
2 Parameters
```

Each help source file can contain several modules. The Librarian recognizes an individual module as a group of key-1 and subkey lines, and their associated message text. A module is terminated either by another key-1 line or an end-of-file (EOF) record.

The format of a help source file is:

```
1 key-1 name
   .
   .
   .
  help message text
   .
   .
   .
2 key-2 name
   .
   .
   .
  help message text
   .
   .
   .
n key-n name
   .
   .
   .
1 key-1 name
```

The Librarian stores the key-1 name in its module name table; therefore, the name of the module is the same as the key-1 name. The subsequent numbers in the first column indicate that the line is a subkey. A module can have several subkeys with the same number. For example, a help module describing a command might have the following key-2 lines:

```
2 parameters
2 arguments
```

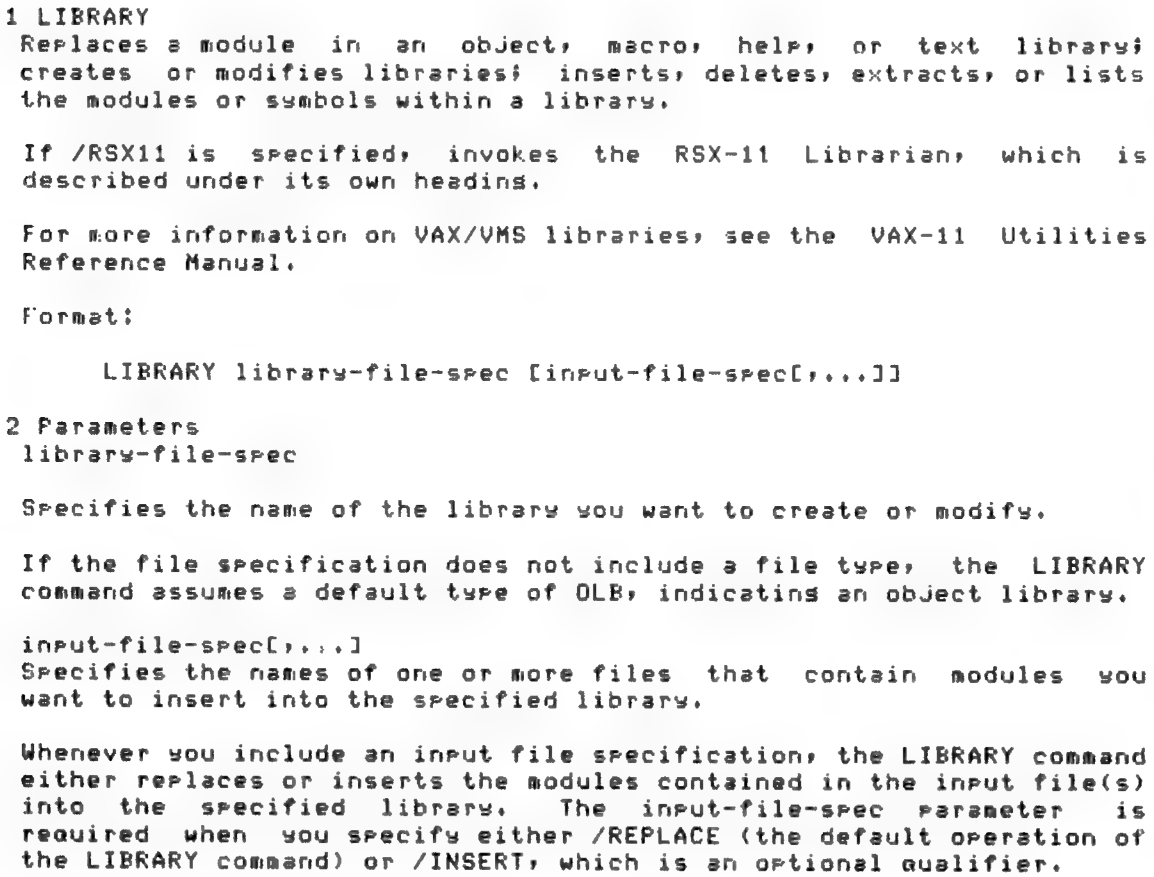
You can insert comments anywhere in a module. When the Librarian encounters an exclamation mark as the first character on a line, it assumes that the line is for comments. Comment lines that follow a key-1 line are included in the module. However, when your program retrieves help text, the Librarian does not output the comment lines.

The text of the help message may be any length; the only restriction to the text is that it cannot contain a number or a slash (/) in the first column of any line. A number in the first column of a line indicates that the line is a key. A slash (/) in the first column indicates a qualifier line.

A qualifier line is similar to a key line, except that the Librarian returns a list of all the qualifier lines when you request help either on a key-1 or on the key containing the qualifiers (usually a key-2 named "Qualifiers"). Therefore, if your help module describes a command that has qualifiers, the Librarian will provide a list of all the command's qualifiers whenever you request help on the command.

10.3.3 Help Message Example

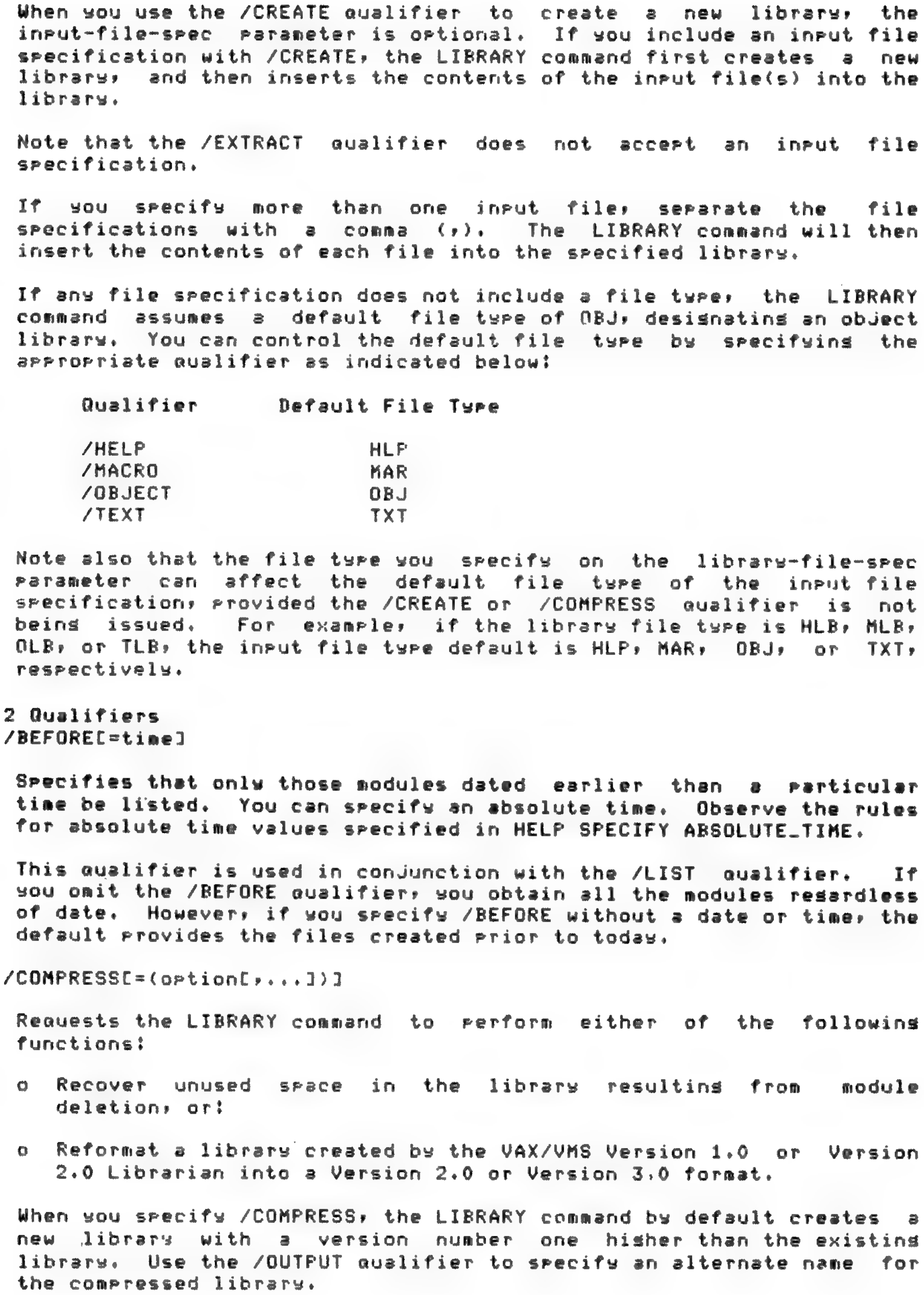
The help module in Figure 10-1 shows the organization of help messages for the DCL LIBRARY command.



```
1 LIBRARY
 Replaces a module  in  an  object, macro, help, or  text  library;
 creates  or modifies libraries; inserts, deletes, extracts, or lists
 the modules or symbols within a library.

 If /RSX11 is  specified,  invokes  the  RSX-11  Librarian,  which  is
 described under its own heading.

 For more information on VAX/VMS libraries, see the  VAX-11  Utilities
 Reference Manual.

 Format:

      LIBRARY library-file-spec [input-file-spec[,...]]

2 Parameters
 library-file-spec

 Specifies the name of the library you want to create or modify.

 If the file specification does not include a file type,  the  LIBRARY
 command assumes a default type of OLB, indicating an object library.

 input-file-spec[,...]
 Specifies the names of one or more files  that  contain  modules  you
 want to insert into the specified library.

 Whenever you include an input file specification, the LIBRARY command
 either replaces or inserts the modules contained in the input file(s)
 into  the  specified  library.  The  input-file-spec  parameter   is
 required  when  you specify either /REPLACE (the default operation of
 the LIBRARY command) or /INSERT, which is an optional qualifier.
```

Figure 10-1: Help Messages for LIBRARY Command

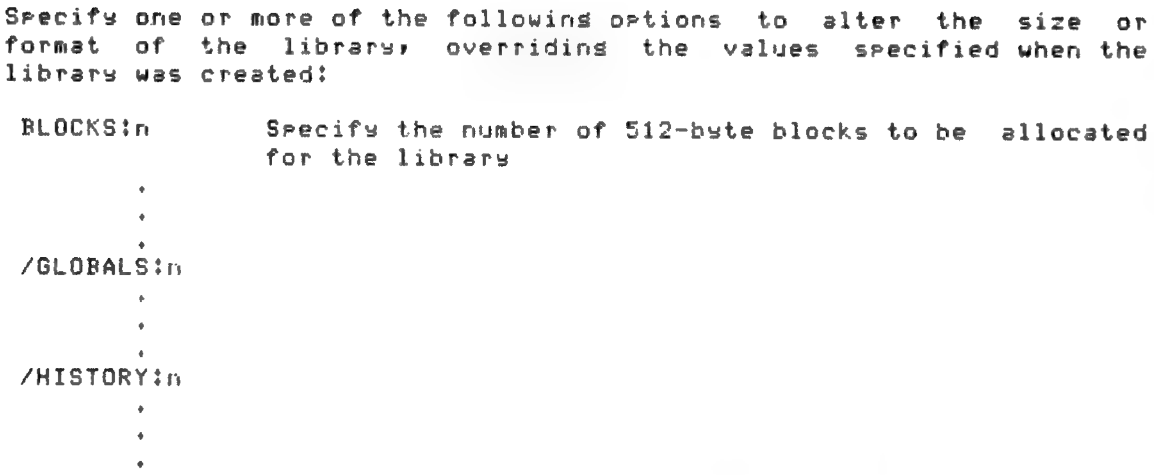
(continued on next page)



```
 When you use the /CREATE qualifier  to  create  a  new  library,  the
 input-file-spec  parameter is optional.  If you include an input file
 specification with /CREATE, the LIBRARY command first creates  a  new
 library,  and then inserts the contents of the input file(s) into the
 library.

 Note that the /EXTRACT  qualifier  does  not  accept  an  input  file
 specification.

 If  you  specify  more  than  one  input  file,  separate  the   file
 specifications  with  a  comma  (,).   The  LIBRARY command will then
 insert the contents of each file into the specified library.

 If any file specification does not include a file type,  the  LIBRARY
 command  assumes  a  default  file type of OBJ, designating an object
 library.  You can control the default file  type  by  specifying  the
 appropriate qualifier as indicated below:

      Qualifier      Default File Type

      /HELP                HLP
      /MACRO               MAR
      /OBJECT              OBJ
      /TEXT                TXT

 Note also that the file type you  specify  on  the  library-file-spec
 parameter  can  affect  the  default  file  type  of  the  input file
 specification, provided the /CREATE or  /COMPRESS  qualifier  is  not
 being  issued.   For  example,  if the library file type is HLB, MLB,
 OLB, or TLB, the input file type default is HLP, MAR,  OBJ,  or  TXT,
 respectively.

2 Qualifiers
/BEFORE[=time]

 Specifies that only those modules dated  earlier  than  a  particular
 time be listed.  You can specify an absolute time.  Observe the rules
 for absolute time values specified in HELP SPECIFY ABSOLUTE_TIME.

 This qualifier is used in conjunction with the /LIST  qualifier.   If
 you omit the /BEFORE qualifier, you obtain all the modules regardless
 of date.  However, if you specify /BEFORE without a date or time, the
 default provides the files created prior to today.

/COMPRESS[=(option[,...])]

 Requests the LIBRARY command  to  perform  either  of  the  following
 functions:

 o  Recover  unused  space  in  the  library  resulting  from   module
    deletion, or:

 o  Reformat a library created by the VAX/VMS Version 1.0  or  Version
    2.0 Librarian into a Version 2.0 or Version 3.0 format.

 When you specify /COMPRESS, the LIBRARY command by default creates  a
 new  library  with  a  version  number  one  higher than the existing
 library.  Use the /OUTPUT qualifier to specify an alternate name  for
 the compressed library.
```

Figure 10-1 (Cont.): Help Messages for LIBRARY Command

(continued on next page)



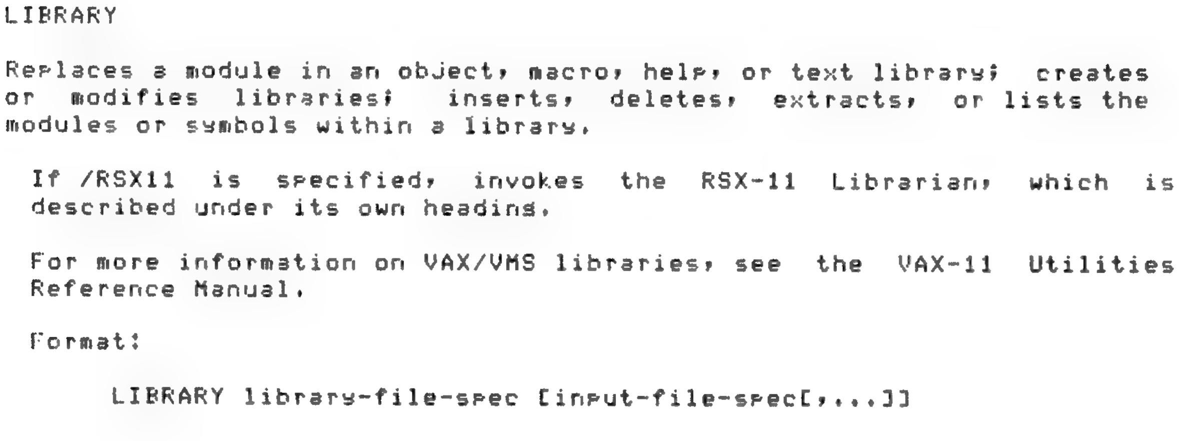
```
Specify one or more of the following options to alter the size or
format of the library, overriding the values specified when the
library was created:

 BLOCKS:n        Specify the number of 512-byte blocks to be allocated
                 for the library
       .
       .
       .
 /GLOBALS:n
       .
       .
       .
 /HISTORY:n
       .
       .
       .
```

Figure 10-1 (Cont.): Help Messages for LIBRARY Command

VAX/VMS provides two help retrieval routines: LBR$GET_HELP and LBR$OUTPUT_HELP. LBR$GET_HELP (see Section 10.5.7) returns help text from a specified help library. LBR$OUTPUT_HELP (see Section 10.5.15) outputs help text to a user-supplied help routine. (Default library searching and interactive help prompts are also described in Section 10.5.15.)

When you retrieve help messages, you specify the key-1 level, followed by any subkeys that contain appropriate help information. The Librarian returns the help message associated with the key path you specified.

To retrieve the LIBRARY command's key-1 help information, you would type the DCL command HELP LIBRARY. The Librarian would return the associated help message, followed by the message, "Additional information available:" and a list of all the key-2 names in the module. In this case, the Librarian also returns a list of all the qualifiers specified in the qualifier lines. Figure 10-2 displays the message returned from the HELP LIBRARY command.



```
LIBRARY

Replaces a module in an object, macro, help, or text library; creates
or modifies libraries; inserts, deletes, extracts, or lists the
modules or symbols within a library.

  If /RSX11 is specified, invokes the RSX-11 Librarian, which is
  described under its own heading.

  For more information on VAX/VMS libraries, see the VAX-11 Utilities
  Reference Manual.

  Format:

       LIBRARY library-file-spec [input-file-spec[,...]]
```

Figure 10-2: HELP LIBRARY Display

(continued on next page)

```
Additional information available:

Parameters Qualifiers

/BEFORE[=time]    /COMPRESS[=(OPTION[,...])]     /CREATE[=option[,...])]
/CROSS-REFERENCE[=(option],...])]                  /DELETE=(module[,...])
/EXTRACT=(module[,...]) /FULL  /GLOBALS (D)  /NOGLOBALS  /HELP  /HISTORY
/INSERT  /LIST[=file-spec]  /NOLIST (D)  /LOG  /NOLOG (D)  /MACRO /NAMES
/NONAMES (D)    /OBJECT (D)    /ONLY=(module[,...])    /OUTPUT=file-spec
/REMOVE=(symbol[,...])  /REPLACE (D)  /SELECTIVE-SEARCH  /SINCE  [=TIME]
/SQUEEZE (D) /NOSQUEEZE /TEXT /WIDTH=n /MODULE=module-name /RSX11
```

Figure 10-2 (Cont.): HELP LIBRARY Display

Note that you could not retrieve the key-2 level, "parameters," by typing HELP PARAMETERS. The Librarian searches for a subkey only after successfully finding the higher-level keys. In other words, if you want to retrieve a key-3 message, you would have to specify the key-1 and key-2 lines that are associated with the key-3 line.

Note also that if you request help information on the /GLOBALS qualifier, the Librarian will return /NOGLOBALS as well. As shown in Figure 10-2, you can provide information on a qualifier that has more than one form by associating two qualifier lines with a single help message.

When the Librarian successfully searches the key path to the requested key, it displays all the key names in that path, followed by the help message associated with the last specified key. For example:

```
$HELP LIBRARY/HELP

         LIBRARY

             /HELP
              Indicates that the library is a help library.  When you specify
              the  /HELP qualifier, the library file type defaults to HLB and
              the input file type defaults to HLP.

               For information on how to create help  files,  see  the  VAX-11
               Utilities Reference Manual.
```

If you try to retrieve a help message that does not have a corresponding key in the module name table, the Librarian will issue a message. For example:

```
$HELP FIRE

         Sorry, no documentation on FIRE


         Additional information available:
```

This message will be followed by a list of all the module names in the module name table.

If you have correctly specified the key-1 line, but have requested a subkey that does not exist, the Librarian will print a message. For example:

```
        $HELP LIBRARY/FIRE

           Sorry, no documentation on LIBRARY/FIRE

           Additional information available:

           Parameters  Qualifiers
           /COMPRESS[=(option[,...])]   /CREATE[=(option[,...])]
                .
                .
                .
```

The message will include a list of all the subkeys associated with the last correctly specified key.

The help library serves as the repository for all your help messages. You can include help messages in your programs by calling the appropriate Librarian routines described in the next section.

10.4 SHAREABLE IMAGE LIBRARIES

A shareable image library is made up of the symbol tables of shareable images, which serve as input to the VAX-11 Linker. To create a shareable image library, use the LIBRARY command with the /SHARE qualifier, as follows:

```
     $ LIBRARY/CREATE/SHARE MYSHARLIB MYHRIMG1,MYSHRIMG2,SHRIMG3
```

The library is then specified in the LINK command exactly as if it were an object library:

```
     $ LINK/MAP/FULL MYPROG, MYSHARLIB/LIBRARY
```

The /INCLUDE qualifier is used to explicitly include a shareable image:

```
     $ LINK/MAP/FULL MYPROG, MYSHARLIB/INCLUDE=(MYSHR1)/LIBRARY
```

For each shareable image found that either contains a necessary symbol or was specifically requested with the /INCLUDE qualifier, the linker looks up the image file module (default file string is .EXE) and processes it as if it had been specified in an options file.

Unless disabled with the /NOSYSSHR qualifier, the linker also searches the library SYS$LIBRARY:IMAGELIB.OLB after processing any user default libraries (LNK$LIBRARY). Modules found in IMAGELIB.OLB are opened with a default file string of SYS$LIBRARY:.EXE.

The default file type for the LIBRARY/SHARE command is OLB for the shareable image symbol table library and EXE for the input files.

The librarian uses the GSMATCH of the shareable image as the module ident in the library. The linker issues a warning message if the GSMATCH of the library module is not equal to the GSMATCH found in the corresponding shareable image. A warning message is also issued if the creation date/times found in the library and the shareable image differ.

Users should note that a module inserted into a shareable image library contains only the module header and end of module record, which are extracted from the global symbol table of the input shareable image. Consequently, although it is not an illegal action, there is little reason to extract modules from a shareable image library.

10.5 LIBRARIAN ROUTINES

The Librarian provides a set of 25 routines that your programs can call to:

- Initialize a library
- Open a library
- Look up a key in a library
- Insert a new key in a library
- Return the names of the keys
- Delete a key and its associated text
- Read text records
- Write text records

Your programs can call the Librarian routines using the VAX-11 standard calling sequence provided in all languages that produce VAX-11 native-mode instructions. When your program calls the Librarian routines, it must furnish whatever arguments the routine requires. When the routine completes execution, it returns control to your program. Your program should then analyze the success or failure of the requested operation. See Section 10.6 for an example of calling Librarian routines from a program.

When you link programs that contain calls to Librarian routines, the Linker will automatically locate them during its default search of SYS$SHARE:LBRSHR. (If specified, the LINK command qualifier /NOSYSSHR will inhibit this search.)

See Section 10.6 for an example of linking a program that references the Librarian routines.

For detailed information on linker option files, see the VAX-11 Linker Reference Manual.

Table 10-2 lists each Librarian routine and its function.

The following sections describe, in detail, each of the Librarian routines. The routines appear in alphabetical order. The order in which you call them in your program depends upon the library operations you need to perform. However, in all cases, you must call LBR$INI_CONTROL, followed by LBR$OPEN, before calling any other routine.

Each description of a routine provides the general format for calling the routine from your program. Spaces between arguments are included for readability, and are not part of the syntax. Following the format is a description of each of the arguments.

Each section lists the possible return status codes for the specific routine, with an explanation of the code. Success codes appear alphabetically before an alphabetical listing of the warning and error codes. For more information on return status codes, see the VAX/VMS System Messages and Recovery Procedures Manual.

In addition, each section provides further information about the routine, under "Notes," including specific information about arguments. Any information that does not appear in another category appears under "Notes."

Table 10-2: Librarian Routines

Routine name	Function
LBR$CLOSE	Closes an open library
LBR$DELETE_DATA	Deletes all the text records associated with a specified module
LBR$DELETE_KEY	Deletes a key from a library index
LBR$FIND	Looks up a key by its record identification in preparation for reading the keys associated text
LBR$FLUSH	Writes the contents of modified blocks to the library file and returns the virtual memory that contained those blocks
LBR$GET_HEADER	Retrieves information from the library header
LBR$GET_HELP	Retrieves help text from a specified library
LBR$GET_HISTORY	Retrieves library update history records and calls a user-supplied routine with each record returned
LBR$GET_INDEX	Calls a user-supplied routine to return the contents of an index optionally qualified by a key
LBR$GET_RECORD	Reads a text record associated with a specified key
LBR$INI_CONTROL	Initializes a library index for use by all other routines
LBR$INSERT_KEY	Inserts a new key in the current library index
LBR$LOOKUP_KEY	Looks up a key in the current index in preparation for reading the key's associated text
LBR$OPEN	Opens an existing library or creates a new one
LBR$OUTPUT_HELP	Retrieves help text from an explicitly named library or from user-supplied default libraries and optionally prompts the user for additional help queries

(continued on next page)

Table 10-2 (Cont.): Librarian Routines

Routine name	Function
LBR$PUT_END	Terminates a sequence of records written with LBR$PUT_RECORD
LBR$PUT_HISTORY	Inserts a library update history record
LBR$PUT_RECORD	Writes a text record to be associated with a specified key
LBR$REPLACE_KEY	Replaces an existing key in the current library index
LBR$RET_RMSSTV	Returns the last VAX-11 RMS status value
LBR$SEARCH	Finds index keys that point to specified text
LBR$SET_INDEX	Sets the index number to be used during processing of the library
LBR$SET_LOCATE	Sets Librarian subroutine record access to locate mode
LBR$SET_MODULE	Reads, and optionally updates, a module header
LBR$SET_MOVE	Sets Librarian subroutine record access to move mode

LBR$CLOSE

10.5.1 LBR$CLOSE - Close a Library

The LBR$CLOSE routine closes an open library.

Format

```
LBR$CLOSE (library-index)
```

library-index

A pointer to a longword that contains the library index returned by the LBR$INI_CONTROL routine. The library must be open.

Return Status

LBR$_LIBNOTOPN

The specified library is not open.

LBR$_ILLCTL

The specified library index is not valid.

Notes

If the library index is 0, LBR$CLOSE immediately returns with success.

Upon successful completion, LBR$CLOSE closes the open library, and deallocates all of the memory used for processing the library.

10.5.2 LBR$DELETE_DATA - Delete Text Records

The LBR$DELETE_DATA routine deletes all the text records associated with the specified module.

Format

LBR$DELETE_DATA (library-index, txtrfa)

library-index

A pointer to a longword that contains the library index returned by the LBR$INI_CONTROL routine. The library must be open.

txtrfa

A pointer to a 2-longword array that contains the record's file address (RFA) of the text you want to delete.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_INVRFA

The specified RFA is not valid.

LBR$_LIBNOTOPN

The specified library is not open.

LBR$_STILLKEYS

Keys in other indexes still point at the text; therefore, the specified text was not deleted.

Notes

If the reference count of the text is 0 (there are no other indexes pointing at the text), LBR$DELETE_DATA will delete the specified text records. If the reference count is not 0 (there are keys in other indexes pointing at the text), the Librarian returns the error LBR$_STILLKEYS.

The Librarian reuses data blocks that contain no text.

LBR$DELETE_KEY

10.5.3 LBR$DELETE_KEY - Delete a Key

The LBR$DELETE_KEY routine deletes a key from a library index.

Format

```
LBR$DELETE_KEY (library-index, key-name)
```

library-index

A pointer to a longword that contains the index returned by the LBR$INI_CONTROL routine. The library must be open.

key-name

A longword that contains one of the following:

1. The value of the key (for libraries with binary keys)
2. The address of a string descriptor for the key (for libraries with ASCII keys)

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_KEYNOTFND

The specified key has not been found.

LBR$_LIBNOTOPN

The specified library is not open.

LBR$_UPDURTRAV

The specified index update is not valid as an embedded routine.

Notes

If LBR$DELETE_KEY finds the key specified by key-name in the current index, it deletes the key.

You cannot call LBR$DELETE_KEY within the user-supplied routine specified in either the LBR$SEARCH or LBR$GET_INDEX routines.

10.5.4 LBR$FIND - Lookup a Key by its RFA

The LBR$FIND routine looks up a library key by its record's file address (RFA) and prepares to read the key's associated text.

Format

```
LBR$FIND (library-index, txtrfa)
```

library-index

A pointer to a longword that contains the library index returned by the LBR$INI_CONTROL routine. The library must be open.

txtrfa

A pointer to a 2-longword array that contains the RFA returned by the LBR$LOOKUP_KEY routine.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_ILLIDXNUM

The specified index number is not valid.

LBR$_INVRFA

The specified RFA is not valid.

LBR$_LIBNOTOPN

The specified library is not open.

Notes

If the specified RFA is valid, LBR$FIND initializes internal tables so that you can read the associated text.

LBR$FLUSH

10.5.5 LBR$FLUSH

The LBR$FLUSH routine writes modified blocks to the library file and returns the virtual memory in which they were stored.

Format

LBR$FLUSH (library-index, block-type)

library-index

The address of a longword that contains the library index returned by the LBR$INI_CONTROL routine. The library must be open.

block-type

If LBR$C_FLUSHDATA is specified, the data blocks will be flushed. If LBR$C_FLUSHALL is specified, first the data blocks and then the index will be written.

Return Status

LBR$_BADPARAM

A value passed to the LBR$FLUSH routine was either out of range or an illegal value.

LBR$_NORMAL

The operation completed successfully.

LBR$_WRITERR

An error occurred during the writing of the cached updated blocks to the library file.

Notes

LBR$FLUSH cannot be called from other Librarian routines that reference cache addresses, or by routines called by Librarian routines.

10.5.6 LBR$GET_HEADER - Retrieve Library Header Information

The LBR$GET_HEADER routine returns information from library's header to the caller.

Format

```
LBR$GET_HEADER (library-index, retary)
```

library-index

> The address of a longword that contains the library index returned by the LBR$INI_CONTROL routine. The library must be open.

retary

> An array of 128 longwords that receives the library header. The information in the returned array is shown in Table 10-3. The symbols are defined by the $LHIDEF macro in SYS$LIBRARY:STARLET.MBL.

Table 10-3: Library Header Information Array Offsets

Offset in Longwords	Symbolic Name	Contents
0	LHI$L_TYPE	Library type
1	LHI$L_NINDEX	Number of indexes
2	LHI$L_MAJORID	Library format major identification
3	LHI$L_MINORID	Library format minor identification
4	LHI$T_LBRVER	ASCIC version of Librarian
12	LHI$L_CREDAT	Creation date/time
14	LHI$L_UPDTIM	Date/time of last update
16	LHI$L_UPDHIS	Reserved to DIGITAL (returns 0)
17	LHI$L_FREEVBN	First logically deleted block
18	LHI$L_FREEBLK	Number of deleted blocks
19	LHI$B_NEXTRFA	Record's File Address (RFA) of end of library
21	LHI$L_NEXTVBN	Next VBN to allocate at end of file
22	LHI$L_FREIDXBLK	Number of free preallocated index blocks

(continued on next page)

Table 10-3 (Cont.): Library Header Information Array Offsets

Offset in Longwords	Symbolic Name	Contents
23	LHI$L_FREEIDX	Listhead for preallocated index blocks
24	LHI$L_HIPREAL	VBN of highest preallocated block
25	LHI$L_IDXBLKS	Number of index blocks in use
26	LHI$L_IDXCNT	Number of index entries (total)
27	LHI$L_MODCNT	Number of entries in index 1 (module names)
28	LHI$L_MHDUSZ	Number of bytes of additional information reserved in module header
29	LHI$L_MAXLUHREC	Maximum number of library update history records maintained
30	LHI$L_NUMLUHREC	Number of library update history records in history
31	LHI$L_LIBSTATUS	Library status (false if there was an error closing the library)
32-128		Reserved to DIGITAL

Return Status

LBR$_LIBNOTOPN

The specified library is not open.

LBR$_ILLCTL

The specified library index is not valid.

Notes

Upon successful completion, LBR$GET_HEADER places the library header information into the array.

10.5.7 LBR$GET_HELP - Return Help Text

LBR$GET_HELP returns help text in a help library to the calling program. This routine is called by LBR$OUTPUT_HELP. Most application programs will use LBR$OUTPUT_HELP, which is more powerful and easier to use.

Format:

```
LBR$GET_HELP (library-index, [line-width], [routine], [data],
key-1, key-2,...,key-n)
```

library-index

A pointer to a longword that contains the index returned by the LBR$INI_CONTROL routine. The library must be open.

line-width

The address of a longword that contains the width of the listing line.

routine

The address of a specified routine to call for each line of text you want output.

data

The address of a longword of data to pass to the routine specified in the routine argument.

key-1,key-2,...,key-n

The address(es) of one or more string descriptors for the key(s) that define the text to be output.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_LIBNOTOPN

The specified library is not open.

LBR$_NOTHLPLIB

The specified library is not a help library.

Notes

The optional line-width argument controls the width of the listing line when available help topics are printed. If you do not supply a line-width, or if you specify 0, the line-width defaults to 80 characters per line.

If you do not supply a routine argument, LBR$GET_HELP calls the Run-Time Library procedure LIB$PUT_OUTPUT to send the help text lines to the current output device (SYS$OUTPUT). However, if you want SYS$OUTPUT for your program to be a disk file, rather than the terminal, it is recommended that you supply a routine to output the text.

If the key-1 descriptor is 0, or if it is not present, LBR$GET_HELP will assume that the key-1 name is "HELP," and it ignores all the other keys. For key-2 through key-n, a descriptor address of 0, or a length of 0, or a string address of 0 will terminate the list.

The key argument may contain any of the following special character strings:

String	Meaning
*	Return all first-level help text in the library
KEY...	Return all help text associated with the specified key and its subkeys
*...	Return all help text in the library

LBR$GET_HELP returns all help text in the same format as the output returned by the DCL command HELP. If you do not want the help text indented to the appropriate help level, you must supply your own routine to change the format.

The routine that you specify to output each help text line contains an argument list of four longwords:

1. The first argument contains the address of a string descriptor for the line to be output.

2. The second argument contains the address of a longword that points to one or more flag bits. The flags describe the contents of the text being passed. The possible flags are:

   HLP$M_NOHLPTXT - The specified help text cannot be found.
   HLP$M_KEYNAMLIN - The text contains the key names of the printed text.
   HLP$M_OTHERINFO - The text is part of the information provided on additional help available.

   (The $HLPDEF macro in SYS$LIBRARY:STARLET.MLB defines these flag symbols.)

   Note that if no flag bit is set, help text is being passed.

3. The third argument is the address specified in the data argument in LBR$GET_HELP (or the address of a 0 constant if no argument has been supplied).

4. The fourth argument is the address of a longword containing the current key level.

The routine that you specify must return with success or failure status. A failure status (low bit = 0) terminates the current call to LBR$GET_HELP.

10.5.8 LBR$GET_HISTORY

The LBR$GET_HISTORY routine returns each Library Update History record to a user-specified action routine.

Format:

```
     LBR$GET_HISTORY (control-index, action-routine)
```

control-index

The library index returned from the LBR$INI_CONTROL routine.

action-routine

A user-supplied routine that is called for each Library Update History record. A longword containing the address of a descriptor for the buffer containing a copy of the record is passed to the routine.

Return Status

LBR$_EMPTYHIST

The history is empty.

LBR$_INTRNLERR

An internal Librarian error occurred.

LBR$_NOHISTORY

This library does not have an update history.

LBR$_NORMAL

A normal exit from the routine occurred.

LBR$GET_INDEX

10.5.9 LBR$GET_INDEX - Return the Contents of an Index

LBR$GET_INDEX calls a user-supplied routine to return the contents of an index optionally qualified by a key.

Format:

```
     LBR$GET_INDEX   (library-index,   index-number,   routine-name,
     [match-desc])
```

library-index

A pointer to a longword that contains the index returned by the LBR$INI_CONTROL routine. The library must be open.

index-number

A pointer to a longword that contains the number of the primary index you want to return.

routine-name

The name of a routine that you supply to be called for each element in the index.

match-desc

The address of a string descriptor that identifies selected entries.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_ILLIDXNUM

The specified index number is not valid.

LBR$_LIBNOTOPN

The specified library is not open.

LBR$_NULIDX

The specified index is empty.

Notes

LBR$GET_INDEX calls with two arguments the routine you specified in the routine-name argument. The two arguments are:

1. Either the address of a string descriptor for the entry (for libraries with ASCII keys) or the address of a value (for libraries with binary keys).

2. The address of a 2-longword array containing the entry's RFA.

If the routine returns a false value (low bit = 0), LBR$GET_INDEX stops searching the index.

Note that the string descriptor passed to your routine is valid only for the duration of the supplied routine. If you need to use the string descriptor in later processing, you must first copy the string.

If you include the match-desc argument, LBR$GET_INDEX supplies only entries that match the specified string. The string can contain embedded asterisks (*) and percent signs (%) that serve as wild card characters in the string description. If you do not supply the match-desc argument, LBR$GET_INDEX uses an asterisk and matches all entries. The match-desc argument is supported only in libraries with ASCII keys.

The routine that you specify in routine-name cannot contain calls to either the LBR$DELETE or LBR$INSERT_KEY routine.

LBR$GET_RECORD

10.5.10 LBR$GET_RECORD - Read a Text Record

The LBR$GET_RECORD routine returns the next text record associated with a key.

Format

```
LBR$GET_RECORD (library-index, inbufdes [, outbufdes])
```

library-index

A pointer to a longword that contains the index returned by the LBR$INI_CONTROL routine. The library must be open and the LBR$LOOKUP_KEY routine must have been called.

inbufdes

A pointer to a string descriptor for the user-supplied buffer.

outbufdes

A pointer to a string descriptor for the actual record returned. This parameter must be specified when Librarian subroutine record access is set to locate mode. The Notes section below contains a description of the locate and move modes.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_LIBNOTOPN

The specified library is not open.

LBR$_LKPNOTDON

The requested key lookup has not been done.

RMS$_EOF

An attempt has been made to read past the logical end of text.

Notes

Before calling LBR$GET_RECORD, you must first find the key by calling LBR$LOOKUP_KEY or LBR$FIND. When you call LBR$GET_RECORD, the Librarian fills the input buffer (described by inbufdes) with the text record. If you have optionally specified the output buffer string descriptor (outbufdes), the Librarian fills it with the actual length and address of the data.

LBR$GET_RECORD uses two record access modes: locate mode and move mode. Move mode is the default. The LBR$SET_LOCATE and LBR$SET_MOVE subroutines set these modes, as described in Sections 10.5.23 and 10.5.25, respectively. The record access modes are mutually exclusive, that is, when one is set the other is turned off. If move mode is set, LBR$GET_RECORD copies the record to a user-specified buffer. If locate mode is set, LBR$GET_RECORD returns a descriptor to the record by way of an internal subroutine buffer by using the descriptor address specified by the outbufdes parameter. The second parameter, inbufdes, is not used when locate mode is set.

LBR$INI_CONTROL

10.5.11 LBR$INI_CONTROL - Initialize a Library Index

The LBR$INI_CONTROL routine initializes a library index for use by all other Librarian routines. You must call this routine before calling any other Librarian routine.

Format

    LBR$INI_CONTROL (library-index, func, [type, namblk])

library-index

    A pointer to a longword that will receive the index for the library.

func

    The address of a longword that contains the library function to be performed. Valid functions are LBRC_CREATE, LBRC_READ, and LBR$C_UPDATE. (These symbols are defined by the $LBRDEF macro in SYS$LIBRARY:STARLET.MLB.)

type

    The address of a longword that contains the library type. If you specify a library type, LBR$OPEN will check for the correct library type.

namblk

    The address of a VAX-11 Record Management Services (VAX-11 RMS) NAM block. If the NAM block has a file identification in it because it was used before, the Librarian will use the VAX-11 RMS open-by-NAM block option. The Librarian will fill in the information in the NAM block so that it can be used at a later time to open the library. This argument is optional and should be used if the library will be opened many times during a single run of the program. For a detailed description of VAX-11 RMS NAM blocks, see the VAX-11 Record Management Services Reference Manual.

Return Status

LBR$_NORMAL

    The library index was initialized successfully.

LBR$_ILLFUNC

    The function requested is not valid.

LBR$_ILLTYP

    The specified library type is not valid.

LBR$_TOOMNYLIB

    An attempt was made to allocate more than 16 control indexes; 16 is the maximum allowed.

Notes:

After you initialize the library index, you must open the library, or create a new one using the LBR$OPEN routine. You can then call other Librarian routines that you need. Once you have completed working with a library, close it with the LBR$CLOSE routine.

LBR$INI_CONTROL initializes a library by filling the longword referenced by the library-index argument with the index of the library. Upon completion of the call, the index can be used to refer to the current library in all future routine calls. Therefore, your program must not alter this value.

LBR$INSERT_KEY

10.5.12 LBR$INSERT_KEY - Insert a New Key

The LBR$INSERT_KEY routine inserts a new key in the current library index.

Format

```
LBR$INSERT_KEY (library-index, key-name, txtrfa)
```

library-index

A pointer to a longword that contains the library index returned by the LBR$INI_CONTROL routine. The library must be open.

key-name

A longword that contains one of the following:

1. The address of the value of the key (for libraries with binary keys)

2. The address of a string descriptor for the key (for libraries with ASCII keys)

txtrfa

A pointer to a 2-longword array that contains the RFA of the associated text. You must use the RFA returned by the first call to the LBR$PUT_RECORD routine.

Return Status

LBR$_IDXFUL

The specified index is full.

LBR$_ILLCTL

The specified library index is not valid.

LBR$_INVRFA

The specified RFA does not point to valid text.

LBR$_KEYINDEX

The index already contains the specified key.

LBR$_LIBNOTOPN

The specified library is not open.

LBR$_UPDURTRAV

The specified index update is not valid as an embedded routine.

Notes

You cannot call LBR$INSERT_KEY within the user-supplied routine specified in either the LBR$SEARCH or LBR$GET_INDEX routines.

10.5.13 LBR$LOOKUP_KEY - Look Up a Library Key

The LBR$LOOKUP_KEY routine looks up a key in the library's current index and prepares to read the text associated with the key.

Format

```
LBR$LOOKUP_KEY (library-index, key-name, txtrfa)
```

library-index

A pointer to a longword that contains the index returned by the LBR$INI_CONTROL routine. The library must be open.

key-name

A longword that contains one of the following:

1. The address of the value of the key (for libraries with binary keys)

2. The address of a string descriptor for the key (for libraries with ASCII keys)

txtrfa

A pointer to a 2-longword array that receives the RFA of the text you want to read.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_KEYNOTFND

The specified key was not found.

LBR$_LIBNOTOPN

The specified library is not open.

Notes

If LBR$LOOKUP_KEY finds the specified key, it initializes internal tables so that you can read the associated text.

The Librarian returns the RFA (consisting of the VBN and the byte offset) to the 2-longword array pointed to by txtrfa. Note that the array contains an RFA of only 48 bits.

LBR$OPEN

10.5.14 LBR$OPEN - Open a Library

The LBR$OPEN routine opens an existing library or creates a new one. This routine must be called after you call LBR$INI_CONTROL and before you call any other Librarian routine.

Format

```
LBR$OPEN (library-index [,fns, create-options, dns, rlfna, rns,
rnslen])
```

library-index

A pointer to a longword that contains the index returned by LBR$INI_CONTROL.

fns

The address of a string descriptor for the file name string. This argument must be included unless the VAX-11 RMS NAM block address was previously supplied in the LBR$INI_CONTROL routine and it contained a file identification. Otherwise, an error (LBR$_NOFILNAM) will result.

create-options

If you are creating a library with LBR$C_CREATE, you must include the create-options argument. The create-options argument is an array of 20 longwords that describes the characteristics of the library you want to create. Table 10-4 shows the entries that the array must contain. (The $LBRDEF macro in SYS$LIBRARY:STARLET.MLB defines the symbols listed.)

dns

The address of a string descriptor for the default file name string.

rlfna

The address of a VAX-11 RMS NAM block for the related file name. If you do not specify rlfna, no related file name processing occurs. See the VAX-11 Record Management Services Reference Manual for details on processing related file names.

rns

The address of a string descriptor for the resultant file name string. If an error occurs during an attempt to open the library, the expanded name string will be returned.

rnslen

A pointer to a longword that receives the length of the resultant file name string (or the length of the expanded name string if there was an error in opening the library).

Table 10-4: Create-Options Array

Offset in Longwords	Symbolic Name[1]	Contents
0	CRE$L_TYPE	Library type
	LBR$C_TYP_UNK (0)	Unknown/unspecified
	LBR$C_TYP_OBJ (1)	Object and/or shareable image
	LBR$C_TYP_MLB (2)	Macro
	LBR$C_TYP_HLP (3)	Help
	LBR$C_TYP_TXT (4)	Text
	(5-127)	Reserved to DIGITAL
	LBR$C_TYP_USR (128-255)	User-defined
1	CRE$L_KEYLEN 0	32-bit unsigned keys
	non-0	Maximum length of ASCII keys
2	CRE$L_ALLOC non-0	Initial library file allocation
3	CRE$L_IDXMAX	Number of primary indexes (maximum of 8)
4	CRE$L_UHDMAX	Number of additional bytes to reserve in module header
5	CRE$L_ENTALL	Number of index entries to pre-allocate
6	CRE$L_LUHMAX	Maximum number of library update history records to maintain
7	CRE$L_VERTYP	Format of library to create:
	CRE$C_VMSV2	VAX/VMS Version 2.0
	CRE$C_VMSV3	VAX/VMS Version 3.0
8 below):	CRE$L_IDXOPT	Index option (see Notes
	CRE$C_HLPCASING	Treat character case as it is for help libraries

[1] Defined by the $LBRDEF macro in SYS$LIBRARY:STARLET.MLB.

(continued on next page)

Table 10-4 (Cont.): Create-Options Array

Offset in Longwords	Symbolic Name[1]	Contents
	CRE$C_OBJCASING	Treat character case as it is for object libraries
	CRE$C_MACTXTCAS	Treat character case as it is for macro and text libraries
9-20		Reserved to DIGITAL

[1] Defined by the $LBRDEF macro in SYS$LIBRARY:STARLET.MLB.

Return Status

LBR$_ERRCLOSE

When the library was last modified while opened for write access, the write operation was interrupted. This left the library in an inconsistent state.

LBR$_OLDLIBRARY

The specified Version 1.0 library has been opened.

LBR$_ILLCREOPT

The requested create options are not valid or not supplied.

LBR$_ILLCTL

The specified library index is not valid.

LBR$_ILLFMT

The specified library format is not valid.

LBR$_ILLFUNC

The specified library function is not valid.

LBR$_INSVIRMEM

No virtual memory is available for the specified function.

LBR$_LIBOPN

The specified library is already open.

LBR$_NOFILNAM

The fns argument was not supplied, or the VAX-11 RMS NAM block was not filled in.

LBR$_OLDMISMCH

The library function requested conflicts with the old library type specified.

LBR$_TYPMISMCH

The library type requested conflicts with the library type specified.

Notes

When the library is successfully opened, the Librarian reads the library header into memory, sets the default index to 1, and updates the library index.

If the library cannot be opened because it is already open for a write operation, LBR$OPEN will retry the open operation every one second for a maximum of 30 seconds before returning the VAX-11 RMS error, RMS$_FLK, to the caller.

The input of uppercase and lowercase characters is treated differently for help, object, macro, and text libraries:

- Help libraries -- Index keywords are in the format in which they were entered, that is, individual uppercase and lowercase characters are entered that way. A second, identically spelled keyword (but of a different or mixed character case) to the same library replaces the previous keyword. However, character case is ignored for match operations; for example, "help Sample" and "help SAMPLE" will access the same module.

- Object libraries -- Index keywords are in the format in which they were entered. A second, identical keyword (but of a different or mixed character case) to the same library initiates a separate keyword; the previous keyword is not replaced. Match operations require that the character case be identical; for example, for SAMPLE, the user must enter SAMPLE, not Sample or sample.

- Text and macro libraries -- All index keywords are converted to uppercase characters; for example, Sample becomes SAMPLE. Likewise for match operations, either Sample or sample matches SAMPLE.

LBR$OUTPUT_HELP

10.5.15 LBR$OUTPUT_HELP - Outputs Help Text

The LBR$OUTPUT_HELP routine outputs help text to a user-specified output routine. The text is obtained from an explicitly named help library, or optionally, from user-specified default help libraries. An optional prompting mode is available that enables LBR$OUTPUT_HELP to interact with a user and continue to provide help information after the initial help request has been satisfied.

Format

```
     LBR$OUTPUT_HELP (output-routine [, [output-width] [, [line-desc]
                    [, [library-name] [,[flags] [,[input-routine]]]]]])
```

The input-routine parameter must be specified when in prompting mode, that is, when HLP$M_PROMPT is specified in the flags parameter. (The input-routine parameter is not required in the absence of the flags parameter.)

output-routine

The address of a routine to call, with the string descriptor of a text string, for each line of text that is output. You should specify either the address of LIB$PUT_OUTPUT or a routine of your own that has the same calling format as LIB$PUT_OUTPUT.

output-width

The address of a longword that contains the width of the text line to be passed to the user-supplied output routine. If output-width is omitted or 0, the default output-width is 80 characters per line.

line-desc

The address of a string descriptor for a string that contains the one or more help keys that define the help requested, for example, the HELP command line minus the HELP command and qualifiers. The default is a string descriptor for an empty string.

library-name

The address of a string descriptor for the main library name string. Null by default; uses default help libraries. If the device and directory specifications are omitted, the default is SYS$HELP. The default file type is HLB.

flags

The address of an unsigned longword that contains the following flags:

HLP$M_PROMPT	Set when the HELP command should be run in prompting mode.
HLP$M_PROCESS	Set when default library searching should be in effect for process default help libraries.
HLP$M_GROUP	Set when default library searching should be in effect for group default help libraries.

HLP$M_SYSTEM	Set when default library searching should be in effect for system default help libraries.
HLP$M_LIBLIST	Set when the list of default libraries available should be output with the list of topics available.
HLP$M_HELP	Precede the list of topics available in a help library with the major portion of the help file text on HELP.

(The $HLPDEF macro in SYS$LIBRARY:STARLET.MLB defines these flag symbols.)

If this address is omitted, the default is for prompting and all default library searching to be enabled, but no library list will be generated and no help text will precede the list of topics.

input-routine

The address of a routine to call, with the string descriptor of a prompt string, for each input record required by LBR$OUTPUT_HELP. You should specify either the address of LIB$GET_INPUT or a routine of your own that has the same calling format as LIB$GET_INPUT. This argument must be supplied when the HELP command is run in prompting mode (HLP$M_PROMPT is set or defaulted).

Return Status

LBR$_ILLINROU

The input routine was improperly specified or omitted.

LBR$_ILLOUTROU

The output routine was improperly specified or omitted.

LBR$_TOOMNYARG

Too many arguments were specified.

Notes

LBR$OUTPUT_HELP accepts help keys in the same format as LBR$GET_HELP (see Section 10.5.7), with the following qualifications:

1. If the keyword HELP is supplied, help text on HELP is output, followed by a list of HELP subtopics available.

   If no help keys are provided, or the line-desc argument is 0, a list of topics available in the root library is output.

2. If the line-desc argument contains a list of help keys, then each key must be separated from its predecessor by a slash (/) or by one or more spaces.

3. The first key can specify a library to replace the main library as the root library in which LBR$OUTPUT_HELP searches for help. A key used for this purpose must have the form <@filespec>, where filespec is subject to the same restrictions as the library-name argument. If

the library specified is an enabled user-defined default library, then filespec can be abbreviated as any unique substring of that default library's logical name translation.

In default library searches, you can define one or more default libraries for LBR$OUTPUT_HELP to search for help information not contained in the root library. You do this by equating the logical names HLP$LIBRARY, HLP$LIBRARY_1,...,HLP$LIBRARY_999 to the file specifications of the default help libraries. These logical names can be defined in the process, group, or system logical name tables.

If default library searching is enabled by the flags argument, LBR$OUTPUT_HELP uses those flags to determine which logical name tables are enabled, and then automatically searches any user default libraries that have been defined in those logical name tables. The library search order proceeds as follows: root library, main library (if specified and different from the root library), process libraries (if enabled), group libraries (if enabled), system libraries (if enabled). If the requested help information is not found in any of these libraries, LBR$OUTPUT_HELP returns to the root library and issues a help not found message.

To enter an interactive help session (after your initial request for help has been satisfied) the HLP$M_PROMPT bit in the flags argument must be set.

You can encounter four different types of prompts in an interactive help session. Each type represents a different level in the hierarchy of help available to you:

1. If the root library is the main library and you are not currently examining help for a particular topic, the prompt "Topic?" is output.

2. If the root library is a library other than the main library and you are not currently examining help for a particular topic, a prompt of the form "@<library-spec>Topic?" is output.

3. If you are currently examining help for a particular topic (and subtopics), a prompt of the form "<keyword...>subtopic?" is output.

4. A combination of 2 and 3.

When you encounter one of these prompt messages, you can respond in any one of several ways. Each type of response, and its effect on LBR$OUTPUT_HELP in each prompting situation, is described below:

Response	Action in the Current Prompt Environment[1]
keyword [,...]	(1,2) Search all enabled libraries for these keys. (3,4) Search additional help for the current topic (and subtopic) for these keys.

[1] Keyed to the prompt in the preceding list.

Response	Action in the Current Prompt Environment[1]
@filespec [keyword[,...]]	(1,2) Same as above, except that the root library is the library specified by filespec. If the specified library does not exist, treat @filespec as a normal key. (3,4) Same as above; treat @filespec as a normal key.
?	(1,2) Display a list of topics available in the root library. (3,4) Display a list of subtopics of the current topic (and subtopics) for which help exists.
Carriage Return	(1) Exit from LBR$OUTPUT_HELP. (2) Change root library to main library. (3,4) Strip the last keyword from a list of keys defining the current topic (and subtopic) environment.
CTRL/Z	(1,2,3,4) Exit from LBR$OUTPUT_HELP.

[1] Keyed to the prompt in the preceding list.

LBR$PUT_END

10.5.16 LBR$PUT_END - Terminate a Text Sequence Written to a Library

The LBR$PUT_END routine terminates a text sequence written to a library by the LBR$PUT_RECORD routine.

Format

LBR$PUT_END (library-index)

library-index

A pointer to a longword that contains the index returned by the LBR$INI_CONTROL routine. The library must be open.

Returns Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_LIBNOTOPN

The specified library is not open.

Notes

Call LBR$PUT_END after you have written text records to the library with the LBR$PUT_RECORD routine. LBR$PUT_END terminates a text sequence by attaching a 3-byte logical end-of-file record (hexadecimal 77,00,77) to the text.

10.5.17 LBR$PUT_HISTORY

The LBR$PUT_HISTORY routine adds an update history record to the end of the update history list. If the list is full, the oldest record is deleted before the new record is added.

Format

LBR$PUT_HISTORY (library-index, record-desc)

library-index

The address of a longword that contains the library index returned by the LBR$INI_CONTROL routine.

record-desc

The address of a string descriptor for the record to be added to the library update history.

Return Status

LBR$_INTRNLERR

An internal Librarian error occurred.

LBR$_NOHISTORY

This library does not have an update history.

LBR$_NORMAL

A normal exit from the routine occurred.

LBR$_RECLNG

The record length was greater than that specified by LBR$C_MAXLUHLEN. The record was not inserted or truncated.

LBR$PUT_RECORD

10.5.18 LBR$PUT_RECORD - Write a Text Record

The LBR$PUT_RECORD routine writes a text record beginning at the next free location in the library.

Format

LBR$PUT_RECORD (library-index, bufdes, txtrfa)

library-index

A pointer to a longword that contains the index returned by the LBR$INI_CONTROL routine. The library must be open.

bufdes

A pointer to a string descriptor that contains the buffer to receive the record output.

txtrfa

A pointer to a 2-longword array that receives the RFA of the newly created module header.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_LIBNOTOPN

The specified library is not open.

Notes

If this is the first call to LBR$PUT_RECORD, the Librarian first writes a module header and returns its RFA to the 2-longword array pointed to by txtrfa. LBR$PUT_RECORD then writes the supplied text record to the library.

10.5.19 LBR$REPLACE_KEY - Change Text Pointer or Insert New Key

The LBR$REPLACE_KEY routine inserts a new key in an index by changing the pointer associated with an existing key, or by inserting a new key.

Format

```
LBR$REPLACE_KEY (library-index, key-name, oldrfa, newrfa)
```

library-index

A pointer to a longword that contains the index returned by the LBR$INI_CONTROL routine. The library must be open.

key-name

A longword that contains one of the following:

1. The address of the key's value (for libraries with binary keys)
2. The address of a string descriptor for the key (for libraries with ASCII keys)

oldrfa

A pointer to a 2-longword array that contains the RFA of the old text.

newrfa

A pointer to a 2-longword array that contains the RFA of the new text.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_LIBNOTOPN

The specified library is not open.

LBR$_INVRFA

The specified RFA is not valid.

Notes

If LBR$REPLACE_KEY does not find the key in the current index, it calls the LBR$INSERT_KEY routine to insert the key.

If LBR$REPLACE_KEY finds the specified key, it performs the following:

1. Decreases by 1 the reference count for the old text (pointed to by oldrfa)

2. Increases by 1 the reference count for the new text (pointed to by newrfa)

3. Modifies the entry for the key so that it now points to the new text

10.5.20 LBR$RET_RMSSTV - Return VAX-11 RMS Status Value

The LBR$RET_RMSSTV routine returns the status value from the last VAX-11 RMS function performed by any Librarian subroutine.

Format

```
LBR$RET_RMSSTV
```

This routine takes no parameters.

LBR$SEARCH

10.5.21 LBR$SEARCH - Search an Index

The LBR$SEARCH routine finds index keys that point to the specified text.

Format

```
     LBR$SEARCH (library-index, index-number, rfa-to-find, routine-name)
```

library-index

A pointer to a longword that contains the index returned by the LBR$INI_CONTROL routine. The library must be open.

index-number

A pointer to a longword that contains the number of the primary index you want to search.

rfa-to-find

A pointer to a 2-longword array that contains the RFA of the key you want to find.

routine-name

The name of a routine that you supply to call for each key containing the matching RFA.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_ILLIDXNUM

The specified index number is not valid.

LBR$_KEYNOTFND

The Librarian did not find any keys with the specified RFA.

LBR$_LIBNOTOPN

The specified library is not open.

Notes

Use LBR$SEARCH to find index keys that point to some specified text. For example, you can call LBR$SEARCH to find all the global symbols associated with an object module in an object library.

If LBR$SEARCH finds an index key, it calls a user-supplied routine with two arguments. The two arguments are:

1. Either the address of a string descriptor for an ASCII key or the address of the value of a binary key

2. Address of a 2-longword array that points to the RFA of the associated text

If the specified routine returns a false value (low bit = 0), then the index search terminates.

Note that the key-name argument is valid only for the duration of the call to the user-supplied routine. If you want to use the key-name argument later, you must copy it.

The routine that you specify in routine-name cannot contain any calls to either the LBR$DELETE or LBR$INSERT_KEY routine.

LBR$SET_INDEX

10.5.22 LBR$SET_INDEX - Set the Primary Index Number

The LBR$SET_INDEX routine sets the index number to use during processing of libraries that have more than one index.

Format

```
LBR$SET_INDEX (library-index, index-number)
```

library-index

A pointer to a longword that contains the library index returned by the LBR$INI_CONTROL routine. The library must be open.

index-number

A pointer to a longword that contains the number of the index you want to set.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_ILLIDXNUM

The index number specified is not valid.

LBR$_LIBNOTOPN

The specified library is not open.

Notes

You call LBR$SET_INDEX when working with libraries that contain more than one index. Macro, help, and text libraries contain only one index; therefore, you do not need to call LBR$SET_INDEX. Object libraries contain two indexes. If you want to access the global symbol table, you must call the LBR$SET_INDEX routine to set the index number. User-developed libraries can contain more than one index; therefore, you may need to call LBR$SET_INDEX to set the index number.

Upon successful completion, LBR$SET_INDEX sets the current index to the requested index number. The Librarian numbers indexes starting with 1.

10.5.23 LBR$SET_LOCATE - Set Record Access to Locate Mode

The LBR$SET_LOCATE routine sets the record access of Librarian subroutines to locate mode. If locate mode is set, LBR$GET_RECORD will not copy the requested record to the specified user buffer. Instead, it returns a descriptor of the record in an internal Librarian subroutine buffer.

Format

```
     LBR$SET_LOCATE (library-index)
```

library-index

The address of a longword that contains the index returned by the LBR$INI_CONTROL routine.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_LIBNOTOPN

The specified library is not open.

LBR$SET_MODULE

10.5.24 LBR$SET_MODULE - Read or Update a Module Header

The LBR$SET_MODULE routine reads, and optionally updates, the module header associated with a given record's file address (RFA).

Format

LBR$SET_MODULE (library-index, rfa, [bufdesc,buflen,updatedesc])

library-index

The address of a longword that contains the library index returned by the LBR$INI_CONTROL routine. The library must be open.

rfa

A pointer to the RFA associated with the module header. The Librarian returns the RFA as a result of either the first call to the LBR$PUT_RECORD routine, or a previous call to the LBR$LOOKUP_KEY routine. For a description of record access by RFA, see the VAX-11 Record Management Services Reference Manual.

bufdesc

A pointer to a string descriptor for the buffer that receives the module header.

buflen

The address of a longword to contain the length of the returned module header.

updatedesc

A pointer to a string descriptor of additional data that the Librarian stores with the module header. If you include this argument, the Librarian will update the module header with the additional information.

Return Status

LBR$_HDRTRUNC

The buffer supplied to hold the module header was too small.

LBR$_ILLCTL

The specified library index is not valid.

LBR$_ILLOP

The updatedesc argument was supplied and the library was a Version 1.0 library or the library was opened only for read access.

LBR$_INSVIRMEM

No virtual memory is available for the specified function.

The specified RFA does not point to a valid module header.

LBR$_LIBNOTOPN

The specified library is not open.

Notes

If you specify bufdesc, the Librarian will return the module header into the buffer. If you specify buflen, the Librarian will also return the buffer's length. If you specify updatedesc, the Librarian will update the header information.

You define the maximum length of the update information when you create the library. The Librarian will zero-fill the information if it is less than the maximum length, or truncate the information if it exceeds the maximum length.

LBR$SET_MOVE

10.5.25 LBR$SET_MOVE - Set Record Access to Move Mode

The LBR$SET_MOVE routine sets the record access of Librarian subroutines to move mode. If move mode is set, LBR$GET_RECORD will copy the requested record to the specified user buffer.

Format

```
     LBR$SET_MOVE (library-index)
```

library-index

The address of a longword that contains the index returned by the LBR$INI_CONTROL routine.

Return Status

LBR$_ILLCTL

The specified library index is not valid.

LBR$_LIBNOTOPN

The specified library is not open.

10.6 EXAMPLE OF LIBRARIAN ROUTINES

This section describes LBRDEMO, a VAX-11 FORTRAN program example that illustrates the use of the Librarian routines. LBRDEMO contains calls to the Librarian routines that perform the following operations:

- Name and initialize a text library
- Open or create a text library
- Replace or insert text modules
- Extract text modules
- List the help topics in the system help library
- Retrieve help text from the system help library

The VAX-11 MACRO source module DEMOMAC and the sample program LBRDEMO are shown below. DEMOMAC initializes a FORTRAN COMMON block to allow LBRDEMO to access symbols unavailable to FORTRAN. The circled numbers in the program are keyed to the descriptions that follow.

```
        .TITLE  demo_mac
        .IDENT  'V02-000'


;
; Macros
;
        $credef                  ; Define create options array offsets
        $dscdef                  ; Define string descriptor offsets
        $lbrdef                  ; Define librarian parameters
        $lbrctltbl               ; Define library control table offsets
        $namdef                  ; Define NAM block offset
;
; Set up FORTRAN COMMON block to allow FORTRAN main program to ❶
; access librarian data
;
        .PSECT  lbrdata, PIC, OVR, REL, GBL, SHR, NOEXE, RD, WRT, LONG

        .long   lbr$c_read       ; func_read
        .long   lbr$c_create     ; func_create
        .long   lbr$c_update     ; func_update
        .long   lbr$c_typ_txt    ; type_text
        .long   lbr$c_typ_hlp    ; type_help
        .long   rms$_eof         ; rmseof
        .long   dsc$k_class_d    ; class_dynamic
                                 ; offsets into create options array
                                 ; values are divided by 4 to convert byte
                                 ; offsets into longword offsets
        .long   cre$l_type/4     ;  type of library
        .long   cre$l_keylen/4   ;  max key length
        .long   cre$l_alloc/4    ;  initial library disk allocation
        .long   cre$l_idxmax/4   ;  number of indices
        .long   cre$l_uhdmax/4   ;  size of additional module header data
        .long   cre$l_entall/4   ;  number of index entries to preallocate
        .SBTTL  nam_init - Initialize RMS NAM block

;++
;       Initialize array to be an RMS NAM block ❷
;
; Calling sequence:
;
;       call nam_init (nam_array, result_desc)
;
; Inputs:
;
;       nam_array       Address of an array of bytes to be initialized
;                       as a NAM block.
;
;       result_desc     Address of string descriptor for resultant name
;                       string.
;
; Outputs:
;
;       The nam_array is initialized as a NAM block.  The expanded
;       and resultant name strings point to the string described by
;       result_desc.
;
; Routine value:
;
;       Always success
;
;--
```

```
        .PSECT  $code$, PIC, REL, SHR, EXE, RD, NOWRT

        .ENTRY  nam_init,^M<R2, R3, R4, R5, R6>

        movl    4(AP), r6                       ; Get address of NAM block
        movc5   #0, (SP), #0, #nam$c_bln, (r6)  ; Zero the NAM block
        movl    8(AP), R0                       ; Get address of resultant
                                                ; name string descriptor
        $NAM_STORE NAM = R6,-                   ; Initialize the NAM fields
                        BLN = #nam$c_bln,-      ;  block length
                        BID = #nam$c_bid,-      ;  block id
                        RSS = dsc$w_length(R0),- ; resultant name string size
                        ESS = dsc$w_length(R0),- ; expanded name string size
                        RSA = @dsc$a_pointer(R0),-; resultant name string
                                                ; address
                        ESA = @dsc$a_pointer(R0)  ; expanded name string
                                                ; address
        movl    #1,r0                           ; Return with success
        ret

        .END
C       Demo program for library access procedures
C
C       Version V03-000
C
C

        PROGRAM LBR_DEMO

        IMPLICIT INTEGER (A-Z)
        EXTERNAL lib$get_input, list_module, lbr$_illcreopt
C
C       The common block is declared in a MACRO source module to gain
C       access to system definitions not available to FORTRAN.
C

        COMMON /lbrdata/ func_read, func_create, func_update,
        1 type_text, type_help, rmseof, class_dynamic, create_type,
        2 create_keylen, create_alloc, create_idxmax, create_uhdmax,❶
        3 create_entall

        CHARACTER*128 library_name, library_rsn, module_name
        CHARACTER*128 input_line

        BYTE help_namblk (56), string_desc_bytes (8), dyn_desc_bytes (8)

        DIMENSION create_options (0:49), old_module_rfa (2), module_rfa (2),
        1 dyn_string (2), string_desc (2)

C
C       The equivalence of the STRING_DESC array with the STRING_DESC_BYTES
C       array is done to access the string descriptor class field.
C

        EQUIVALENCE (string_desc, string_desc_bytes),
        1 (dyn_string, dyn_desc_bytes)

        library_open = .false.
        have_name = .false.
C
C       Initialize NAM block for use with HELP library.❷
C

        CALL nam_init (help_namblk, library_rsn)
```

```
C
C       Allocate a dynamic string and initialize string descriptors.
C
        dyn_string (1) = 0
        dyn_string (2) = 0
        dyn_desc_bytes (4) = class_dynamic
        string_desc (1) = 0
        string_desc (2) = 0
        string_desc_bytes (4) = class_dynamic
        status =
        1 lib$sget1_dd (2048, string_desc)      !Allocate 2048 byte string
        IF (status) GOTO 100
 10     CALL lib$signal (%VAL (status))
        STOP
C
C       Main dispatch loop -- Get action and dispatch ❸
C
 100    TYPE 9000
        READ (5, *, END=300) action
        GOTO (1000, 2000, 3000, 4000, 5000, 6000, 7000), action + 1
        GOTO 100
 200    CALL lib$signal (%VAL (status))
        GOTO 100
C
C       Close library and exit
C
 300    IF (library_open) THEN
                status = lbr$close (library_index)
                IF (.NOT. status) GOTO 10
                END IF
        STOP

C
C       Give some help
C
 1000   TYPE 9020
        GOTO 100
C
C       Name new library
C
C       If there is a library open, it will be closed.  The new library
C       name is accepted.
C
 2000   IF (library_open) THEN ❹
                status = lbr$close (library_index)
                IF (.NOT. status) CALL lib$signal (%VAL (status))
                library_open = .false.
                END IF
        TYPE 9040
        READ (5, 9110, END=100) name_length, library_name
        library_name = library_name (1:name_length)//'.TLB'
        have_name = .true.
        GOTO 100
C
C       Open or Create a TEXT library ❺
C
 3000   IF (.NOT. have_name) GOTO 8000
        TYPE 9540
        READ (5, *, END = 100) create_flag
        IF (create_flag) THEN
 3100       TYPE 9060
            READ (5, *, END=100) max_key_length ❻
            function = func_create
            create_options (create_type) = type_text
            create_options (create_keylen) = max_key_length
            create_options (create_alloc) = 100
            create_options (create_idxmax) = 1
            create_options (create_uhdmax) = 0
            create_options (create_entall) = 100
            ELSE
            function = func_update       !Opening existing library. Set for
            END IF                       !updates
C
C     Initialize librarian for this library
C
      status = lbr$ini_control (library_index, function, type_text)
      IF (.NOT. status) GOTO 200
C
C     Open or create the library
C
      status = lbr$open (library_index, library_name, create_options)
C
C     If there is an illegal create_option, then it must be the value for
C     the maximum key length. Reprompt.
C
      IF (status .EQ. %LOC (lbr$_illcreopt)) THEN
      CALL lib$signal (lbr$_illcreopt)
      GOTO 3100
      END IF

      IF (.NOT. status) GOTO 200
      library_open = .TRUE.
      GOTO 100
C
C     Insert or replace a module in the text library
C
 4000 IF (.NOT. library_open) GOTO 8020 ❼
      TYPE 9080
      READ (5, 9110, END=100) name_length, module_name
      replacing = lbr$lookup_key (library_index, !Determine if
                                     !replacing or inserting module
      1 module_name (1:name_length), old_module_rfa)
      TYPE 9100
 4100 READ (5, 9110, END = 4200) line_length, input_line
      status = lbr$put_record (library_index,
      1 input_line (1:line_length), module_rfa)
 4120 IF (.NOT. status) CALL lib$signal (%VAL (status))
      GOTO 4100
C
C     Module text has been inserted into the library. Terminate the
C     module.
C
 4200 status = lbr$put_end (library_index)❽
      IF (.NOT. status) CALL lib$signal (%VAL (status))
      status = lbr$replace_key (library_index,   !Insert or replace key
      1 module_name (1:name_length), old_module_rfa, module_rfa)
      IF (.NOT. status) CALL lib$signal (%VAL (status))
      status = .TRUE.
      IF (replacing) status = lbr$delete_data (library_index, !If replac-
                                     !ing, delete old module
      1 old_module_rfa)
      IF (.NOT. status) GOTO 200
      GOTO 100
```

```
C
C       Extract module from library and type on terminal
C
 5000   IF (.NOT. library_open) GOTO 8020 ❾
        TYPE 9400
        READ (5, 9050, END = 100) module_name
        status = lbr$lookup_key (library_index, module_name, module_rfa)
        IF (.NOT. status) GOTO 200
 5100   status = lbr$get_record (library_index, string_desc, dyn_string)
        IF ((.NOT. status) .AND. status .NE. rmseof)
        1 CALL lib$signal (%VAL (status))
        IF (status .EQ. rmseof) GOTO 100
        CALL lib$put_output (dyn_string)
        GOTO 5100
C
C       List contents of index of SYS$HELP:HELPLIB.HLB
C
 6000   status = lbr$ini_control (help_index, func_read, type_help) ❿
        IF (.NOT. status) GOTO 200
        status = lbr$open (help_index, %DESCR ('SYS$HELP:HELPLIB.HLB'))
        IF (.NOT. status) GOTO 200
        status = lbr$get_index (help_index, 1, list_module)
        IF (.NOT. status) CALL lib$signal (%VAL (status))
        status = lbr$close (help_index)
        IF (.NOT. status) GOTO 200
        GOTO 100
C
C       Lookup help text in SYS$HELP:HELPLIB.HLB and display on the
C       terminal.
C
 7000   TYPE 9200
        READ (5, 9110, END = 100) line_length, input_line ⓫
        status = lbr$output_help (lib$put_output, , -
        1 input_line (1:line_length), -
        2 %DESCR ('HELPLIB'), , lib$get_input)
        IF (.NOT. status) CALL lib$signal (%VAL (status))
        GOTO 100
C
C       Error routines
C


C
C       No library name given
C
 8000   TYPE 9500
        GOTO 100
C
C       No library open
C
 8020   TYPE 9520
        GOTO 100
C
C       Format statements
C
 9000   FORMAT (' Action (0 for help): ',$)
 9020   FORMAT (' Commands:',/' 1 - Name library',/,
        1' 2 - Open or Create a text library',/,
        2' 3 - Replace/insert module',/,
        4' 4 - Extract module',/,
        3' 5 - List directory of SYS$HELP:HELPLIB.HLB',/,
        5' 6 - Lookup help text in SYS$HELP:HELPLIB.HLB')
```

```
9040    FORMAT (' New library name: ',$)
9050    FORMAT (A)
9060    FORMAT (' Maximum key length: ',$)
9080    FORMAT (' Module name: ',$)
9100    FORMAT (' Enter text. Terminate with a Control-Z:')
9110    FORMAT (Q, A)
9200    FORMAT (' Enter help keys: ',$)
9400    FORMAT (' Module to extract: ',$)
9500    FORMAT (' No library name given')
9520    FORMAT (' No library open')
9540    FORMAT
        1 (' Open existing library (0) or create new library (1): ',$)
        END

         INTEGER FUNCTION list_module (keyname, keyrfa)

         IMPLICIT INTEGER (A-Z)

         CHARACTER *(*) keyname

         TYPE *,keyname
         list_module = .true.                      !Return success
         RETURN
         END
```

❶ LBRDEMO calls eleven of the Librarian routines. To call these routines, LBRDEMO uses a FORTRAN COMMON block to access symbols that are unavailable to FORTRAN. The symbols specify the functions available for initializing a library and the options available for creating a text library. (See ❻.)

The COMMON block is initialized in the VAX-11 MACRO source module DEMOMAC; the symbols are accessed by linking the two programs together. If required and unless /NOSYSSHR is specified, the linker automatically includes the Librarian routines. To assemble and link the two programs, you can create a command procedure that contains the following commands:

```
$ FORTRAN /LIST LBRDEMO
$ MACRO  /LIST DEMOMAC
$ LINK /EXE=DEMO /MAP=DEMO /FULL LBRDEMO, DEMOMAC
```

This command procedure produces the executable image DEMO.EXE.

❷ DEMOMAC contains a subroutine, NAM_INIT, that initializes an array to be used as a VAX-11 RMS NAM block when retrieving help messages from the system help library. (See ⓫ for information on how to use the NAM block option; for an example of opening a library without the NAM block option, see ❹.)

❸ The main part of LBRDEMO, beginning at line 100, asks what library operation you want to perform. By typing "0" (for help), you request an operation menu that lists the program's six library functions:

1. Name library

2. Open or create a text library

3. Replace/insert module

4. Extract module

5. List directory of SYS$HELP:HELPLIB.HLB

6. Look up help text in SYS$HELP:HELPLIB.HLB

Each operation on the menu references at least one of the Librarian routines. After the Librarian performs the requested operation, it returns you to this menu. At this point, you can either perform another operation by typing the appropriate number, or you can terminate the program by typing CTRL/Z. Typing CTRL/Z performs the following functions:

- Calls the LBR$CLOSE routine (line 300) to close a previously opened text library

- Executes the STOP instruction to terminate the program

❹ The first operation on the menu allows you to name the library you want to access. If a text library has been previously opened, LBRDEMO calls the LBR$CLOSE routine (line 2000) to close the library. LBRDEMO then prompts you for the library name, appends the text library file type TLB to the name you specify, and assigns the string to the symbol LIBRARY_NAME. The variable HAVE_NAME is set to true, and LBRDEMO returns to the operations menu.

❺ The second operation on the menu opens or creates a library using the library name you specify in 1. LBRDEMO asks whether you are opening an existing library or creating a new one (line 3000). If you want to create a library, the variable FUNCTION will take the value stored in FUNC_CREATE; if you are opening a library, the variable FUNCTION will take the value stored in FUNC_UPDATE. In either case, LBRDEMO calls the LBR$INI_CONTROL routine to initialize the text library index.

The three arguments to the LBR$INI_CONTROL routine specify that the library index is named TEXT_INDEX, the library function to be performed is contained in the variable FUNCTION, and the library type is TEXT. The variable FUNCTION is accessed through the COMMON block. Note that the call to LBR$INI_CONTROL initializes a library index without using the NAM block option. Instead, the call to LBR$OPEN opens a library using the specifications contained in the LIBRARY_NAME argument.

Note also that the call to the LBR$INI_CONTROL routine does not enable you to access a library; it merely initializes a library index to which the other Librarian routines can refer. The subsequent call to LBR$OPEN opens the library for access. If you try to perform a library operation before initializing a library index and opening a library file, LBRDEMO will print "No library name given" (line 4000) and will return you to the operations menu.

❻ If you are opening an existing library, the LBR$OPEN routine will open for access the library specified in LIBRARY_NAME. If you are creating a new library, the LBR$OPEN routine will use the information in the CREATE_OPTIONS array to create the library. The symbols in the create options array are accessed through the COMMON block. You must supply the maximum key length of the library; the rest of the create options are predefined by the program.

❼ After you have initialized and opened (or created) a library, you can insert or replace a text module (the third operation on the menu). The method by which the Librarian inserts a module depends on whether the key to the module already exists in the library index.

In the sequence beginning at line 4000, the variable REPLACING is associated with the status of the LBR$LOOKUP_KEY routine. This variable is later used to test whether the module being inserted already exists in the library (see ❽).

LBRDEMO creates the new text module by prompting you to specify the module name and enter the text (line 4100). As you enter the text records, LBRDEMO calls the LBR$PUT_RECORD routine to write the text records to the module you specified. Note that the first call to LBR$PUT_RECORD fills in the MODULE_RFA argument with the address of the module. In this way, the first call to LBR$PUT_RECORD provides a mechanism for accessing the new module in subsequent calls to the Librarian. After the module is written, LBRDEMO calls the LBR$PUT_END routine to terminate the writing sequence.

❽ LBRDEMO inserts the module into the library using the LBR$REPLACE_KEY routine. If the key you are inserting does not already exist in the module name table, LBR$REPLACE_KEY will insert the key into the index. The new key will point to the inserted module.

If the key you are inserting already exists, LBR$REPLACE_KEY must perform a replace operation. In a replace operation, the Librarian changes the index key from the address specified in OLD_MODULE_RFA to the address specified by MODULE_RFA. The key then points to the new module, but the old module still exists physically in the library. Therefore, after the call to LBR$REPLACE_KEY, LBRDEMO must delete the old module.

If the value of REPLACING is true, the call to LBR$LOOKUP_KEY will find a module with the same name and return its address in the location specified by the argument OLD_MODULE_RFA. LBRDEMO then calls the LBR$DELETE_DATA routine to delete physically the old module from the library.

❾ The fourth operation on the menu extracts a module from the currently open library and displays it on the terminal. LBRDEMO first prompts you for the module you want to extract, then calls LBR$LOOKUP_KEY. The call to LBR$LOOKUP_KEY searches TEXT_INDEX for the key specified by the MODULE_NAME argument. If the key is found, LBR$LOOKUP_KEY will fill in the array specified by MODULE_RFA with the record's file address (RFA) of the module you want to read.

Once the module is found, LBRDEMO calls the LBR$GET_RECORD routine to retrieve the text records. The first argument to the routine, STRING_DESC, specifies the input buffer that receives the text record. The second argument, DYN_STRING, is an optional argument that receives the actual length and address of the text record read.

The program sequence beginning at line 5100 constitutes a read loop. As LBR$GET_RECORD returns each text record, LBRDEMO checks for the end-of-file (RMSEOF). If it is not at the end of the file, LBRDEMO will call the Run-Time Library procedure LIB$PUT_OUTPUT to output the text record described by the string descriptor DYN_STRING.

After LBR$GET_RECORD retrieves all the text records in the module, LBRDEMO returns you to the operations menu.

❿ The fifth operation on the menu, beginning at line 6000, lists the contents of the system help library, SYS$HELP:HELPLIB.HLB.

The LBR$INI_CONTROL routine initializes the library index, called HELP_INDEX. The second argument, FUNC_READ, indicates that the library is initialized for reading. Therefore, any attempt to modify the library in subsequent calls will result in an error. The symbol FUNC_READ is accessed through the COMMON block.

The call to LBR$OPEN opens system help library for read access. The second argument to the routine contains the file specification for the library.

The call to LBR$GET_INDEX retrieves the contents of the library index specified by the first argument, HELP_INDEX, which was returned by the preceding call to LBR$INI_CONTROL. The second argument tells the routine to get index number 1. The third argument, LIST_MODULE, is the name of the user-supplied routine (at the end of the program) that LBR$GET_INDEX calls to return the list of index entries.

The final call in this sequence, LBR$CLOSE, closes the index returned by the LBR$INI_CONTROL routine and deallocates all of the space for processing the library.

⓫ The sixth and last operation on the menu retrieves help text from the system help library, SYS$HELP:HELPLIB.HLB. After prompting you for initial help keys (line 7000), LBRDEMO calls the LBR$OUTPUT_HELP routine to interactively provide the user with help information.

After retrieving all help text of interest, you return to the operations menu where you can terminate LBRDEMO by typing CTRL/Z (see ❸).

10.7 MESSAGES FOR LIBRARY COMMAND AND LIBRARIAN ROUTINES

The *VAX/VMS System Messages and Recovery Procedures Manual* lists messages issued by the LIBRARY command and Librarian routines, and provides an explanation and a suggested user action for each message.

CHAPTER 11

MESSAGE UTILITY

The VAX-11 Message Utility allows programmers to control the generation of messages by VAX/VMS.

VAX-11 software products produce messages under different circumstances. Usually, these messages indicate that an error has occurred. However, messages can also indicate other conditions -- for example, that a routine has run successfully, or that a default value has been assigned.

Messages are displayed to the user as a line of alphanumeric codes and text explaining the condition that caused the message to be issued. A message is represented to the VAX-11 processor as a longword called the message code. This 32-bit value can be referred to in programs by means of a global symbol called the message symbol.

The information that appears in the message that users receive, the values that make up the message code, and the characters that make up the message symbol are all defined in files called message source files. VAX/VMS has a file of system message information called SYSMSG.EXE.

You can use your own message files by means of the VAX-11 Message Utility, following these steps:

- Use a text editor to create a source file that specifies the information used in messages, message codes, and message symbols.
- Use the MESSAGE command to compile this source file.
- Link the resulting object module, either by itself or with another object module containing a program.
- Run your program so that the messages are accessed, either directly or through the use of pointers.

You can specify how messages are displayed at run time by using the SET MESSAGE command or by using pointers to message information. These features allow you to suit messages to the requirements of your installation.

This chapter begins with a description of messages, the message code, and the message symbol. It then explains the components of a message source file, and how the file can be compiled and linked. It describes different methods of changing messages at run time, and concludes with a list of the messages issued by the Message utility itself.

11.1 THE FORMAT OF MESSAGES

Messages have the following format:

```
     %FACILITY-L-IDENT, message-text
```

% and ,

The percent sign (%) and comma (,) are included as delimiters if any of the first three fields -- FACILITY, L, or IDENT -- is present.

FACILITY

The abbreviated name of the software product that issued the message. You specify the facility name to be used in your message source file by means of the facility definition described in Section 11.3.1.1. The FACILITY field can contain up to nine characters from the character set described in Section 11.3.1.

You can suppress the appearance of FACILITY for your process by means of the /NOFACILITY qualifier on the DCL command SET MESSAGE, described in Section 11.4.2.

L

An indicator showing the severity level of the condition that caused the message. There are five levels, represented by the following codes:

Code	Function
S	Success
I	Informational
W	Warning
E	Error
F	Fatal or severe

You set the severity level of messages in your message source file using either the severity definition, described in Section 11.3.1.2, or a severity qualifier on the message definition, described in Section 11.3.1.4.

You can suppress the appearance of the severity level indicator for your process by means of the /NOSEVERITY qualifier on the DCL command SET MESSAGE, described in Section 11.4.2.

IDENT

A symbol of up to 15 characters representing the message. Valid characters are described in Section 11.3.1. You specify the symbol in the message definition line of your message source file, described in Section 11.3.1.4.

You can suppress the appearance of the IDENT symbol for your process by means of the /NOIDENTIFICATION qualifier on the DCL command SET MESSAGE, described in Section 11.4.2.

message-text

A brief explanation of the cause of the message. You specify the message text in the message definition line of your message source file, described in Section 11.3.1.4.

If you suppress FACILITY, L, and IDENT, the first character of the message text will be capitalized by the Put Message ($PUTMSG) system service.

You can suppress the appearance of the message text for your process by specifying the /NOTEXT qualifier on the DCL command SET MESSAGE, described in Section 11.4.2.

The message can also include up to 255 formatted-ASCII-output (FAO) arguments; that is, character strings that can be used to display, for example, the instruction at which an error occurred or a value of which the user should be aware. The following sample message includes the file specification as an FAO argument:

```
%TYPE-W-OPENIN, error opening _DB0:[MARCEL]BBBB.FOR; as input
```

11.2 THE MESSAGE CODE AND THE MESSAGE SYMBOL

Messages are formatted by the Put Message ($PUTMSG) system service, described in the *VAX/VMS System Services Reference Manual*. The system service finds the information to use in the message by using a message argument vector. The message argument vector includes a 32-bit value that uniquely identifies the message. This 32-bit value is called the message code.

The message code is made up of the following elements, which are described individually in Section 11.3.1:

- The severity level defined in the severity definition or message definition
- The message number assigned automatically by a message definition or specified with the message number specifier
- The facility number defined in the facility definition
- Internal control flags

Figure 11-1 shows the arrangement of the bits in the message code. The message code is described fully in the *VAX Architecture Handbook* description of condition values.

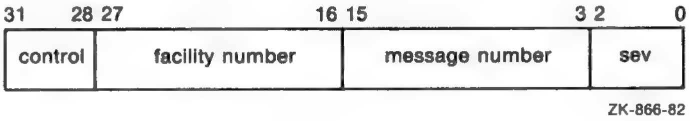


Figure 11-1: Message Code

The message symbol is the symbol that represents the message code. It appears in the object module (the compiled message file) as a global symbol.

The message symbol is constructed of the following elements, described in Section 11.3:

- The symbol prefix defined in the facility definition
- The symbol name defined in the message definition

11.3 CONSTRUCTING MESSAGES

You construct messages by creating a message source file that contains the information that you want to include in the message, the message code, and the message symbol. You then compile the message source file with the Message compiler and link the resulting object module with the VAX-11 Linker. This section describes the contents of the message source file and the use of the compiler and linker to convert the message source file into usable form. It concludes with a sample program that generates messages at run time.

11.3.1 The Message Source File

The message source file contains the information that makes up the message, the message code, and the message symbol. The file is made up of statements that establish the various fields of the message, define symbols, and control the output listing of the file. The message source file has the default file type MSG.

The elements of the message source file, described in the following sections, are:

- Facility definition
- Severity definition
- Message number specifier
- Message definition
- Literal directive
- Identification directive
- Listing directives
- End statement

The format of each statement in the message source file is described in the appropriate section below. A statement in a message source file can take up any number of lines; text that reaches the end of a line and is to be continued on the next line must end with a hyphen (-). The only exceptions to this are the listing title specified with the .TITLE directive and the message text specified in the message definition, which must occupy only one line.

Any line in the message source file can include a comment, delimited by an exclamation point (!). In any line, you can freely insert extra spaces and tabs to improve readability.

Symbols defined in the Message utility can include any of the following characters:

```
A - Z
a - z
1 - 9
$ (dollar sign)
_ (underscore)
```

Expressions used in the Message Utility can include any of the following radix operators to specify the radix of a numeric value:

Operator	Radix	Example
^X	Hexadecimal	^X10
^O	Octal	^O30
^D	Decimal	^D16

The default radix is decimal.

Expressions can include symbols and the unary operators plus sign (+), which assigns a positive value, and minus sign (-), which assigns a negative value. Expressions can also include the following binary operators:

Operator	Function
+	Addition
-	Subtraction
*	Multiplication
/	Division
@	Arithmetic shift

Expressions can include parentheses as special operators. Expressions enclosed in parentheses are evaluated first. Nested parenthetical expressions are evaluated inside to outside.

11.3.1.1 **The Facility Definition** - The texts of messages are grouped in a message source file by the facility (software product) to which they apply, broken down by severity levels. Therefore, a facility definition, specifying the facility to which messages will apply, is the first definition in a message source file. All of the lines following a facility definition apply to that facility, until an end statement or another facility statement is reached.

The facility definition has the format:

```
     .FACILITY[/qualifier,...]     facnam[,]facnum [/qualifier,...]
```

qualifier

One of the qualifiers listed in Table 11-1.

facnam

The facility name, which will be used as the facility field of the message and in the symbol that represents the facility number. It can have up to nine characters. Facility names for VAX/VMS facilities are listed in the VAX/VMS System Messages and Recovery Procedures Manual.

facnum

The facility number, a decimal value in the range of 1 to 2047, or an expression that evaluates to a value in that range. (For information on expressions, see the VAX/VMS Command Language User's Guide.) Facility numbers are usually assigned by the system manager so that no two facilities have the same number. The facility number is used to construct the 32-bit value of the message code.

Note that both the facility name and the facility number are required. They can be separated by a comma or by any number of spaces or tabs.

The facility definition creates a global symbol of the form:

```
     facnam$_FACILITY
```

This symbol can be used to refer to the facility number assigned to the facility.

Table 11-1: Facility Definition Qualifiers

Qualifier	Function
/PREFIX=prefix	Defines an alternate symbol prefix to be used in the message symbol for all messages referring to this facility. The default symbol prefix is the facility name followed by an underscore (_). If /SYSTEM is also specified, the default prefix is the facility name followed by a dollar sign and an underscore ($_). The combined length of the prefix and the message symbol name cannot exceed 31 characters.
/SHARED	Inhibits the setting of the facility-specific bit in the message codes. This qualifier is used only for system service and shared messages. This qualifier is reserved for DIGITAL use.
/SYSTEM	Inhibits the setting of the customer facility bit in the message codes. This qualifier is reserved for DIGITAL use.

11.3.1.2 **The Severity Definition** - Following the facility definition, the message source file generally contains a severity definition specifying the severity level to be associated with the messages that follow. You must include a severity definition if you do not specify the severity individually on each message definition (see Section 11.3.1.4).

The severity definition has the format:

$$\texttt{.SEVERITY} \left\{ \begin{array}{l} \texttt{SUCCESS} \\ \texttt{INFORMATIONAL} \\ \texttt{WARNING} \\ \texttt{ERROR} \\ \texttt{SEVERE} \\ \texttt{FATAL} \end{array} \right\}$$

SEVERE is equivalent to FATAL and can be used interchangeably with it; the severity level code for both of these, as described in Section 11.1, is F.

If you attempt to define a message without specifying a severity level, an error will result. A new facility definition cancels the severity level in effect before it.

11.3.1.3 The Message Number Specifier - The message number is a value used in constructing the message code that represents the message (see Section 11.2). All of the messages following a facility definition are numbered sequentially, beginning with 1 after each facility definition.

In some cases, you may need to supersede this numbering system -- for example, if you want to reserve some message numbers for future assignment. You can specify a message number of your choice using the message number specifier, .BASE, which has the following format:

```
.BASE     number
```

number

A message number to be associated with the next message definition, or an expression that is evaluated as the desired number. This message number is used as a base for the sequential numbering of all messages that follow until another .BASE is encountered or until the end of the messages belonging to the facility.

11.3.1.4 The Message Definition - The message definition specifies the body of the message symbol, the message text, and the number of arguments that can be printed with the message. Any number of message definitions can follow the severity definition. The message definition has the format:

```
name[/qualifier,...]    <message-text>[/qualifier,...]
```

name

This symbol name is combined with the symbol prefix defined in the facility definition to make up the message symbol. The combined length of the prefix and the message symbol name cannot exceed 31 characters.

The symbol name is used in the IDENT field of the message (see Section 11.1) unless the /IDENTIFICATION=name qualifier is specified in the message definition, as described in Table 11-2.

/qualifier

Any of the qualifiers listed in Table 11-2. Qualifiers can be placed before or after the message text, in any order.

message-text

An explanation of the condition that caused the message to be issued. The message text can be delimited either by angle brackets (<>), as shown above, or by quotation marks ("). The text can be up to 255 bytes long; however, you cannot continue the delimited text onto another line. The message text can include directives that insert ASCII strings into the resulting message; these directives are used by the Formatted ASCII Output ($FAO) system service and are described in the VAX/VMS System Services Reference Manual. If you include an FAO directive, you must also use the /FAO_COUNT qualifier, described in Table 11-2.

Table 11-2: Message Definition Qualifiers

Qualifier	Function
/FAO_COUNT=n	Specifies the number of FAO arguments to be included in the message at execution time. (See the VAX/VMS System Services Reference Manual for an explanation of FAO arguments.) The value n must be a decimal number in the range 0 through 255. The $PUTMSG system service uses n to determine how many arguments are to be given to the $FAO system service when constructing the final message text. The default value for n is zero.
/IDENTIFICATION=name	Specifies an alternate character string to be used as the IDENT field of the message (see Section 11.1). The string can include up to nine characters. If this qualifier is not specified, the symbol name defined in the message definition (see above) will be used in the IDENT field of the message.
/USER_VALUE=n	Specifies an optional user value that can be associated with the message. The value n must be a decimal number in the range of 0 through 255. The default is zero. The value can be retrieved by the Get Message ($GETMSG) system service for use in classifying messages by type or by action to be taken.
/SUCCESS /INFORMATIONAL /WARNING /ERROR /SEVERE /FATAL	Specify the severity level to be associated with the message. You can use these qualifiers either to supersede the severity level defined in a severity definition or instead of including severity definitions in your message source file. Only one severity qualifier can be included per message definition.

11.3.1.5 **The Literal Directive** - The literal directive allows you to define global symbols in your message source file. You can either assign values to these symbols or use the default values provided by the statement. The .LITERAL directive has the form:

```
     .LITERAL       symbol[=value][,...]
```

symbol

A symbol name.

value

> Any valid expression. If value is omitted, a default value is assigned. The default value is 1 for the first symbol in the statement; for subsequent symbols in the same statement, the default value is 1 plus the last value assigned.

You can assign default values to a list of symbols. For example:

```
.LITERAL    A,B,C
```

The values of A,B, and C will be 1, 2, and 3.

You can use the .LITERAL directive to define a symbol as the value of another previously defined symbol, or as an expression that results from operations performed on previously defined symbols. In the following example, symbols defined in the facility and message definitions are used to assign values to symbols created with the .LITERAL directive.

```
.FACILITY        SAMPLE,1/PREFIX=MSG$_
.SEVERITY        ERROR
FIRST            <first error>

.
.
.

LAST             <last error>
.LITERAL         LASTMSG=MSG$_LAST

.LITERAL         NUMSG=(MSG$_LAST@-3)-(MSG$_FIRST@-3)    ! # of messages
```

The first .LITERAL directive defines a symbol that has the value of the last 32-bit message code defined. The second .LITERAL directive defines the total number of messages in the source file.

11.3.1.6 **The Identification Directive** - The identification directive allows you to identify the object module produced by the Message Utility. This identification is in addition to the name you assign to the module with .TITLE. You can label the object module by specifying a character string with the directive. The identification directive has the form:

```
.IDENT string
```

string

> A 1- to 31-character string that identifies the module, for example a string that identifies a version number. The string must consist only of alphanumeric characters, underscores, and dollar sign if it is not delimited. If other characters are used, then the string must be delimited with either apostrophes (') or quotation marks (").

Identification strings often consist of version and edit numbers, for example:

```
.IDENT 'VERSION 3.00' !Version and edit number
```

If a MESSAGE source file contains more than one .IDENT, the last directive given establishes the character string that forms part of the object module identification.

11.3.1.7 Listing Directives - You can use two special statements to control the output listing that is produced when you compile your message source file. These statements are the .PAGE directive and the .TITLE directive.

The .PAGE directive enables you to force page breaks in the output listing. It has the format:

```
.PAGE
```

You can only specify one page break with any one .PAGE directive; however, you can use the .PAGE directive as often as you like.

The .TITLE directive enables you to specify the module name and title text that will appear on the top of each page of the listing file. It has the format:

```
.TITLE modname [listing-title]
```

modname

A character string of up to 31 characters that will appear in the object module as the module name.

listing-title

Text to be used as the title of the listing. The text begins with the first nonblank character after the module name through the next 28 characters, which is the maximum length. The listing title cannot be continued onto another line.

11.3.1.8 The End Statement - A group of message definitions is terminated by one of the following: another severity definition, to begin a new group of another severity; an end statement, which terminates the entire list of messages for the facility; or a new facility statement. The end statement has the format:

```
.END
```

A new facility statement performs an implicit .END.

11.3.1.9 Sample Message Source File - The following sample message source file illustrates the various elements described above.

```
.TITLE          SAMPLE Error and Warning Messages
.FACILITY       SAMPLE,1/PREFIX=ABC_
.SEVERITY       ERROR
.IDENT          'VERSION 3.00'

UNRECOG         <Unrecognized keyword !AS>/FAO_COUNT=1
AMBIG           <Ambiguous keyword>

.SEVERITY       WARNING
.BASE           10
SYNTAX          <Invalid syntax in keyword>

.END
```

The messages defined in this message source file belong to a facility with the name SAMPLE and the facility number 1. The first two messages have the severity level E; the third message has the severity level W.

The first message definition above includes the FAO directive !AS (which inserts an ASCII string at the end of the message text) and the corresponding qualifier /FAO_COUNT, as described in Section 11.3.1.4.

The message symbols defined in this message source file are ABC_UNRECOG, ABC_AMBIG, and ABC_SYNTAX. The message numbers are 1, 2, and 10.

11.3.2 Compiling the Message Source File

Message source files must be compiled into object modules before the messages defined in them can be used. You compile your message source file by issuing the MESSAGE command in response to the DIGITAL Command Language (DCL) prompt ($). The MESSAGE command can also be used to create object modules that do not contain message data; instead, they contain pointers to files that contain message data. These pointers are described in Section 11.4.1.

The MESSAGE command has the following format:

```
MESSAGE[/qualifier,...] file-spec[,...]
```

/qualifier

Any of the qualifiers listed in Table 11-3. Note that some qualifiers are mutually exclusive.

file-spec

The message source file to be compiled. If you do not specify a file type, the default is MSG.

You can specify more than one message source file, separated by either commas (,) or plus signs (+). The files will be concatenated and compiled as a single file.

If you specify SYS$INPUT, the message source file(s) must immediately follow the MESSAGE command in the input stream, and both the object module name (given by the /OBJECT qualifier) and the listing file name (given by the /LIST qualifier) must be explicitly stated.

For your convenience, you can put message object modules into object module libraries. These libraries can then be linked with facility object modules.

Table 11-3: MESSAGE Command Qualifiers

Qualifier	Function
/FILE_NAME=file-spec /NOFILE_NAME	Specifies whether the object module contains a pointer to a file containing messages. (Pointers are described in Section 11.4.1.) The default is /NOFILE_NAME, indicating that the object module contains only compiled message information, and no pointers.

(continued on next page)

Table 11-3 (Cont.): MESSAGE Command Qualifiers

Qualifier	Function
	Whenever you specify /FILE_NAME=file-spec, the /NOTEXT qualifier is implied; that is, the /FILENAME and /TEXT qualifiers are mutually exclusive. The /OBJECT qualifier must be in effect, either explicitly or implicitly. The file specification identifies a nonexecutable message file, as explained in Section 11.4.1. The default device and directory for the file specification is SYS$MESSAGE, and the default file type is EXE. No wild card characters are allowed in the file specification.
/LIST[=file-spec] /NOLIST	Controls whether an output listing is created, and optionally provides an output file specification for the listing. When you compile message source files in batch mode, the output listing is created by default. However, in interactive mode, the default is to produce no output listing. The default file type for listing files is LIS. The default device and directory are your default device and directory. No wild card characters are allowed in the file specification.
/OBJECT[=file-spec] /NOOBJECT	Controls whether an object module is created by the message compiler, and optionally provides a file specification for the object module. By default, the compiler creates an object module with the same name as the first message source file and with the file type OBJ. The default device and directory are your default device and directory. No wild card characters are allowed in the file specification.
/SYMBOLS /NOSYMBOLS	Controls whether global symbols will be present in the object module. By default, object modules are created with global symbols. The /SYMBOLS qualifier requires that the /OBJECT qualifier be in effect, either explicitly or implicitly.

(continued on next page)

Table 11-3 (Cont.): MESSAGE Command Qualifiers

Qualifier	Function
/TEXT /NOTEXT	Controls whether the data portion of the object module, containing the information specified in facility, severity, and message definitions, is present in the object module. (Section 11.4.1 describes the use of pointers, which require that data not be present in the object module.) The default is /TEXT. The /TEXT and /FILE_NAME qualifiers are mutually exclusive. The /TEXT qualifier requires that the /OBJECT qualifier be in effect, either explicitly or implicitly. The /NOTEXT qualifier can be used with the /SYMBOLS qualifier to produce an object module containing only global symbols.

11.3.3 Linking the Message Object Module

Before your messages can be used, your program must be linked by the VAX-11 Linker. The VAX-11 Linker resolves symbolic and library references and assigns virtual memory addresses to the relative addresses assigned by the compiler. It produces an executable image file with the file type EXE, which can be run on a VAX-11 processor.

The message object module that results from compiling your message source file can be linked in two different ways. It can be linked with an object module from the facility to which it applies, creating one executable image that contains both the facility code and the message data. Or, it can be linked by itself to create a nonexecutable message file. A nonexecutable message file can be used as a process-permanent message file (see Section 11.4.2) or can be referenced at run time by a pointer (see Section 11.4.1).

Figure 11-2 shows the two ways of linking a message object module.

To link your message object module, issue the DCL command LINK in response to the DCL prompt, as described in the VAX-11 Linker Reference Manual. The following command links the object module COBOLCODE.OBJ and the message object file COBOLMSG.OBJ. The command creates an image map file. The resulting executable image file is named COBOLCODE.EXE.

```
$ LINK/MAP    COBOLCODE,COBOLMSG
```

To link the message object module COBOLMSG.OBJ by itself to create a nonexecutable message file (COBOLMF.EXE), issue the following LINK command:

```
$ LINK/EXECUTABLE=COBOLMF COBOLMSG
```

Note that the Linker will issue a warning message (%LINK-W-USRTFR), which you can ignore, that indicates that the nonexecutable image just created has no user transfer address.

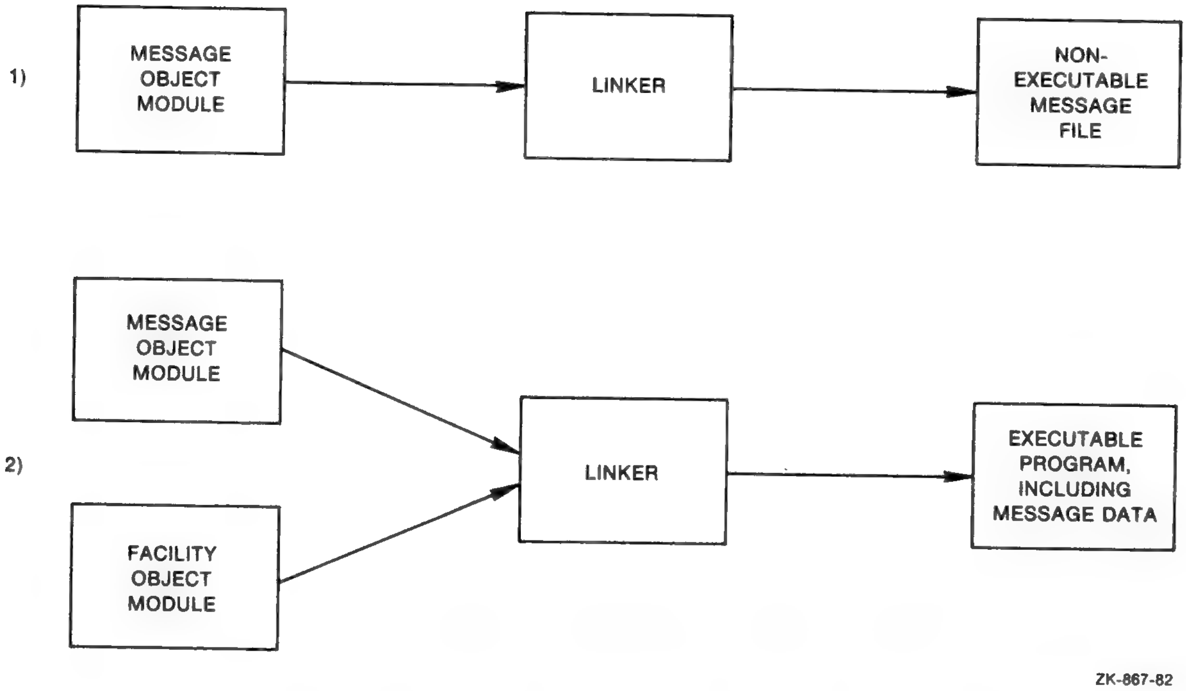


Figure 11-2: Linking a Message Object Module

11.3.4 Running a Program with Messages

This section shows how a program, linked with a message object module, produces messages when run.

The program is a FORTRAN program named TEST.FOR. It contains the following lines:

```
EXTERNAL MSG_SYNTAX,MSG_ERRORS
CALL LIB$SIGNAL(MSG_SYNTAX,%VAL(1),'ABC')
CALL LIB$SIGNAL(MSG_ERRORS)
END
```

This program calls the run-time procedure LIB$SIGNAL, described in the VAX-11 Run-Time Library Reference Manual. The message symbols MSG_SYNTAX and MSG_ERRORS are included as arguments in the procedure calls. The function %VAL is a required FORTRAN compile-time function. The first call also includes the string 'ABC' as an FAO argument.

You compile the FORTRAN program by issuing the following command:

```
$ FORTRAN TEST
```

This command results in an object module named TEST.OBJ.

The message source file, TESTMSG.MSG, contains the following lines:

```
.FACILITY       EFGH,1 /PREFIX=MSG_
.SEVERITY       ERROR
SYNTAX          <Syntax error in string '!AS'>/FAO=1
ERRORS          <Errors encountered during processing>
.END
```

You compile the message source file by issuing the following command:

```
     $ MESSAGE TESTMSG
```

This command results in a message object module named TESTMSG.OBJ.

You link the two object modules by issuing the following command:

```
     $ LINK/NOTRACE TEST+TESTMSG
```

This command results in an executable program named TEST.EXE. You run this program by issuing the following command:

```
     $ RUN TEST
```

The following messages are issued when the program is run:

```
%EFGH-E-SYNTAX, Syntax error in string 'ABC'
%EFGH-E-ERRORS, Errors encountered during processing
```

11.4 CHANGING MESSAGES

Under some circumstances, you may want to change the messages for a facility that runs on your VAX-11 processor. You can make run-time changes on two levels:

1. Per image, by using pointers to message data

2. Per process, by using the DCL command SET MESSAGE

Using pointers is described in Section 11.4.1; using the SET MESSAGE command is described in Section 11.4.2.

11.4.1 Pointers to Message Data

If you have linked your message object module directly with the facility object module, you will have to alter the resulting executable image file to change the message data included in it. This can be time consuming and the resources needed to link the image may not be available. To avoid having to alter the executable image, you can use pointers to a message file instead of linking the message data into the image.

A pointer is created by referring to a nonexecutable message image in a MESSAGE command, using the /FILE_NAME qualifier (described in Section 11.4). The nonexecutable message file is a message source file that has been compiled and linked by itself.

The MESSAGE/FILE_NAME command results in an object module containing only global symbols and the file specification of the message file, which can then be linked with facility object modules.

An object module containing a pointer to message files should have a different file name from the module that actually contains message data.

The following command creates an object module named MESPNTR.OBJ, which contains a pointer to the nonexecutable message file COBOLMF.EXE. (COBOLMF.EXE was created by compiling the message file COBOLMSG.MSG and linking the resulting object module by itself with the qualifier /SHAREABLE=COBOLMF.) Note that it is not necessary to include the file type EXE in the /FILE_NAME qualifier, because EXE is the default. The object module, MESPNTR.OBJ, contains the global symbols defined in the message source file COBOLMSG.MSG.

```
$ MESSAGE/FILE_NAME=COBOLMF /OBJECT=MESPNTR COBOLMSG
```

When the resulting facility image file is run, the message data is retrieved from the message file COBOLMF by the $GETMSG system service. Figure 11-3 illustrates the relationship of the files affected by this command.

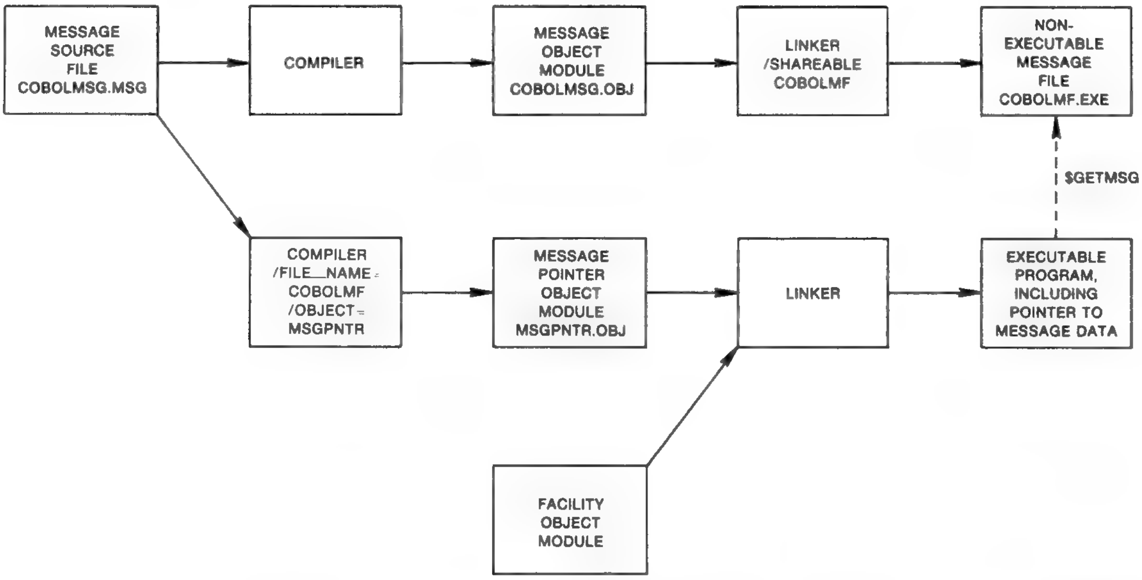

MESSAGE SOURCE FILE COBOLMSG.MSG
COMPILER
MESSAGE OBJECT MODULE COBOLMSG.OBJ
LINKER /SHAREABLE COBOLMF
NON-EXECUTABLE MESSAGE FILE COBOLMF.EXE
$GETMSG
COMPILER /FILE_NAME= COBOLMF /OBJECT= MSGPNTR
MESSAGE POINTER OBJECT MODULE MSGPNTR.OBJ
LINKER
EXECUTABLE PROGRAM, INCLUDING POINTER TO MESSAGE DATA
FACILITY OBJECT MODULE
ZK-868-82

Figure 11-3: Creating a Message Pointer

11.4.2 The SET MESSAGE Command

You can override or supplement the system messages on your system by using the DCL command SET MESSAGE. This command allows you to suppress for your process the various fields of the messages (described in Section 11.1) or to substitute the message data in a nonexecutable message image for the system message data.

The SET MESSAGE command has the following format:

```
SET MESSAGE[/qualifier...]     [file-spec]
```

/qualifier

A qualifier listed in Table 11-4. Multiple command qualifiers can be used, specified in any order.

file-spec

An optional nonexecutable message file. The default file type is EXE; no wild card characters are allowed in the file specification.

The specified message file supersedes any per-process message file already in effect; only one per-process message file can be in effect at any time.

The messages contained in the specified message file are searched by the $GETMSG system service after the per-image messages and before the system-wide messages.

Table 11-4: SET MESSAGE Qualifiers

Qualifier	Function
/DELETE	Removes the process message file from your process. This qualifier cannot be used if you have included a file specification in the SET MESSAGE command; selecting a new per-process message file automatically removes any existing process message file.
/FACILITY /NOFACILITY	Control whether the facility name field (see Section 11.1) is displayed for all messages that occur in your process.
/IDENTIFICATION /NOIDENTIFICATION	Control whether the IDENT field (see Section 11.1) is included for all messages that occur in your process.
/SEVERITY /NOSEVERITY	Control whether the severity level indicator (see Section 11.1) is displayed for all messages that occur in your process.
/TEXT /NOTEXT	Control whether the message text field is displayed for all messages that occur in your process.

Examples

1.
```
$ SET MESSAGE MYMSG
```

   The SET MESSAGE command specifies that the message information in MYMSG.EXE supplements the existing system messages.

2.
```
$ TYPE BBBB.FOR
%TYPE-W-OPENIN, error opening DB1:[MARCEL]BBBB.FOR; as input
-RMS-E-FNF, file not found

$ SET MESSAGE/NOIDENTIFICATION

$
%TYPE-W, error opening DB1:[MARCEL]BBBB.FOR; as input
-RMS-E, file not found
```

   When the first TYPE command is entered, error messages contain all fields. Entering the SET MESSAGE command with the /NOIDENTIFICATION qualifier eliminates the IDENT field from the messages that are subsequently issued.

11.5 MESSAGE UTILITY MESSAGES

The VAX/VMS System Messages and Recovery Procedures Manual lists the error messages issued by the message compiler and provides explanations and suggested user actions for these messages. Most of the user responses entail changing the message source file and reentering the MESSAGE command.

CHAPTER 12

MONITOR UTILITY (MONITOR)

The Monitor Utility (MONITOR) is a system management tool that enables you to obtain information on operating system performance. The Monitor Utility permits you to monitor classes of system-wide performance data (such as system I/O statistics, page management statistics, and time spent in each of the processor modes) at specifiable intervals and to produce a variety of outputs.

This chapter describes MONITOR's features and applications.

The VAX/VMS Monitor Utility enables you to obtain information on operating system performance. It collects system performance data by class and produces three forms of optional output:

- A disk recording file in binary format
- Statistical terminal displays
- A disk file containing statistical summary information in ASCII format

MONITOR collects system performance data from the running system or plays back data recorded previously in a recording file. When you play back data, you can display it, summarize it, and even rerecord it to reduce the amount of data in the recording file. Section 12.5, Basic Modes of Operation, explains these operations in greater detail.

12.1 INVOKING AND TERMINATING MONITOR

The following DCL command invokes the utility:

```
$ MONITOR  class-name  [,...]
```

Each time you issue the DCL command MONITOR, the utility executes a single MONITOR request. Generally, each MONITOR request runs until it completes, and thus terminates at the time specified or implied by the /ENDING qualifier. However, you can press CTRL/C to terminate MONITOR earlier. Pressing CTRL/C terminates MONITOR under normal conditions, ensuring that all files are handled properly.

12.2 COMMAND SUMMARY

This section presents a summary of the MONITOR command format.

12.2.1 Command Format

The MONITOR command adheres to the syntax and grammar rules of all DCL commands. These rules appear in the VAX/VMS Command Language User's Guide. The MONITOR command has the following format:

```
MONITOR  class-name[,...]
```

Command Qualifiers	Defaults
/BEGINNING=time	(see text)
/[NO]COMMENT="string"	/NOCOMMENT
/[NO]DISPLAY[=file-spec]	/DISPLAY=SYS$OUTPUT
/ENDING=time	(see text)
/[NO]INPUT[=file-spec]	/NOINPUT
/INTERVAL=seconds	(see text)
/[NO]RECORD[=file-spec]	/NORECORD
/[NO]SUMMARY[=file-spec]	/NOSUMMARY
/VIEWING_TIME=seconds	(see text)

Class-name Qualifiers	Defaults
/ALL	(see text)
/AVERAGE	(see text)
/[NO]CPU	(see text)
/CURRENT	(see text)
/MAXIMUM	(see text)
/MINIMUM	(see text)
/[NO]PERCENT	(see text)
/TOPBIO	(see text)
/TOPCPU	(see text)
/TOPDIO	(see text)
/TOPFAULT	(see text)

12.2.2 Command Parameters

MONITOR prompts for command parameters as follows:

```
Class(es): class-name[,...]
```

class-name[,...]

Specifies one or more classes of performance data to be monitored. You must specify one or more of the following class names:

DECNET	DECnet-VAX statistics
FCP	File system ACP statistics
IO	System I/O statistics
LOCK	Lock management statistics
MODES	Time spent in each of the processor modes
PAGE	Page management statistics
POOL	Statistics on space allocation in the nonpaged dynamic pool

PROCESSES	Statistics on all processes
STATES	Number of processes in each of the scheduler states

12.2.3 Command Qualifiers

The MONITOR command monitors one or more classes of VAX/VMS performance data. It collects the requested data at a periodic interval specified by the /INTERVAL qualifier. It is capable of producing any combination of three types of output, based on the specification of the /RECORD, /DISPLAY, and /SUMMARY qualifiers. If /RECORD is specified, a binary recording file is produced, containing data collected for requested classes; one record for each class is written per interval. Specifying /DISPLAY produces output in the form of ASCII screen images. Screen images are written at a frequency governed by the /VIEWING_TIME qualifier. If /SUMMARY is specified, an ASCII file is produced, containing summary statistics for all requested classes over the duration of the MONITOR request.

If /INPUT is specified, performance data is collected from a previously created recording file; otherwise, data is collected from counters and data structures of the running system. The MONITOR request begins and ends at times specified by the /BEGINNING and /ENDING qualifiers, respectively. (A MONITOR request can be terminated at any time by pressing CTRL/C.)

/BEGINNING=time

Specifies the time that monitoring begins. You can specify an absolute time, a delta time, or a combination of the two. Observe the syntax rules for time values described in the VAX/VMS Command Language User's Guide.

If you are monitoring a running system and you omit the /BEGINNING qualifier, monitoring begins at the time you issue the MONITOR command. However, if you have also specified the /INPUT qualifier to play back data from an input recording file, /BEGINNING defaults to the beginning time recorded in the input file. If you specify /BEGINNING with a time, but are playing back a recording file, MONITOR selects the later of the beginning time of the file and the beginning time you specify.

If you specify a future time for a request to monitor a running system, MONITOR issues an informational message, and the process issuing the request hibernates until the specified time. This feature can be useful when you run MONITOR from a batch job.

/COMMENT="string"
/NOCOMMENT

Specifies an ASCII string to be stored in the output recording file. The string can contain up to 60 characters. The /COMMENT qualifier is valid only when /RECORD is also specified.

If you omit the qualifier or specify /NOCOMMENT, a string consisting of 60 blanks is stored in the recording file.

/DISPLAY[=file-spec]
/NODISPLAY

Specifies whether information collected by the Monitor Utility is to be displayed as ASCII screen images, and optionally names the disk file to contain the output. If you omit the optional file specification, output is written to the current SYS$OUTPUT device.

By default, display output is produced. Section 12.3.1 describes the display output formats.

/ENDING=time

Specifies the time that monitoring ends. You can specify an absolute time, a delta time, or a combination of the two. Observe the syntax rules for time values described in the VAX/VMS Command Language User's Guide..

If you are monitoring a running system and you omit the /ENDING qualifier, monitoring continues until you terminate the request with CTRL/C. If you have also specified the /INPUT qualifier, so that data is played back from an input recording file, /ENDING defaults to the ending time recorded in the input file. If you specify /ENDING with a time, but are playing back a recording file, MONITOR selects the earlier of the ending time of the file and the ending time you specify.

You can prematurely terminate a request, regardless of the value of the /ENDING qualifier, by pressing CTRL/C. To prematurely terminate a request running in a noninteractive process (that is, a batch job or a detached process or subprocess), issue the appropriate DCL command to terminate the process.

/INPUT[=file-spec]
/NOINPUT

Controls whether performance data is played back from an input file or collected from the running system.

The /INPUT qualifier allows you to specify the name of an input file. No wild card characters are allowed in the file specification. If you omit the file type, the default file type DAT is used. If you omit the file specification, MONITOR assigns the default file name MONITOR.DAT. The current device and directory defaults are applied.

If you omit the qualifier or specify /NOINPUT, performance data is collected from the running system.

/INTERVAL=seconds

Specifies the sampling interval between data collection events, recording events, and display events. Interval values must be in the range of 1 through 9,999,999 seconds.

If you are collecting data from an input file, the specified value is adjusted up to the next whole multiple of the input recording file's interval value (see Section 12.6.3).

For MONITOR operations on a running system, /INTERVAL specifies the number of seconds between successive collection events and thus, the number of seconds between successive recording events. However, the number of seconds between display events is controlled by the /VIEWING_TIME qualifier.

When playing back data from an input file, /INTERVAL does not refer to actual elapsed time, but to the time interval between collection events in the input file. A collection event for an input file occurs once for each recording event of the original MONITOR request. The /INTERVAL qualifier controls the frequency of current recording and display events in terms of the original input interval. For example, suppose collection events for the input file occurred at three-second intervals. If you specify /INTERVAL=6, each current recording and display event includes data from two of the original three-second collection events. If you specify /INTERVAL=8, the value is adjusted up to the next whole multiple of the input recording file's interval, which in this case is nine. Thus, each current recording and display event represents three of the original three-second collection events.

To control the amount of actual time between displays when playing back data from an input file, use the /VIEWING_TIME qualifier.

For /INPUT requests, the interval value defaults to the value specified in the input recording file. The default for monitoring the running system is three seconds.

For more information on specifying the /INTERVAL qualifier for playback, see Section 12.6.3.

/RECORD[=file-spec]
/NORECORD

Controls whether collected data is stored in the specified disk recording file. Note that recording is restricted to files on disks.

The /RECORD qualifier allows you to specify the name of a recording file that will contain the output. No wild card characters are allowed in the file specification. If you omit the file type, the default file type is DAT. If you omit the file specification, output is generated to a file named MONITOR.DAT, in the current default device and directory. If you specify an existing file but omit the version number, a new version of the file is created.

The output consists of all data for the requested classes, regardless of the class-name qualifiers specified.

By default, no recording file output is produced.

/SUMMARY[=file-spec]
/NOSUMMARY

Specifies that an ASCII disk file be created containing summary statistics on all collected data for this request. If the optional file specification is omitted, it defaults to MONITOR.SUM. By default, no summary output is produced.

The summary file, generated at the end of monitoring, contains one page of output for each requested class. The format of each page is similar to that of display output, and is determined by the class-name qualifiers.

Summary output formats are described in Secton 12.3.2.

/VIEWING_TIME=seconds

For /DISPLAY requests, this qualifier specifies the duration for each screen image display. Viewing time values must be in the range of 1 through 9,999,999 seconds.

If you are monitoring the running system, /VIEWING_TIME defaults to the /INTERVAL value. If you specify /INPUT and you are monitoring a recording file, /VIEWING_TIME defaults to three seconds.

12.2.4 Class-name Qualifiers

The class-name qualifiers control the type of display and summary output format generated for each class-name specified. They have no effect on the recording of binary data. Each of these qualifiers applies only to the immediately preceding class-name. Class-name qualifiers must not appear as part of the command verb.

The class-name qualifiers fall into three categories:

- Statistics qualifiers (/ALL, /AVERAGE, /CURRENT, /MAXIMUM, and /MINIMUM) specify which statistics appear in display and summary output. These are conflicting qualifiers; specify no more than one of them with each class-name in a MONITOR request. Note that statistics qualifiers cannot be used with the PROCESSES class-name.

- The data transformation qualifier (/[NO]PERCENT) controls whether data for the selected class-name is expressed as percentages of a whole. This qualifier can only be used with the STATES and MODES class-names.

- Class-specific qualifiers (/CPU, /TOPBIO, /TOPCPU, /TOPDIO, and /TOPFAULT) control the output of a specific class. The /CPU qualifier is used specifically with the MODES class-name to list CPU information for VAX-11/782 attached processor configurations. The remaining qualifiers (referred to generically as /TOP qualifiers) are used with the PROCESSES class-name to produce bar graphs showing the top processes, instead of the standard summary and display output (see Section 12.4.8 for descriptions of these outputs). (Top processes are those that are the heaviest consumers of the resource being monitored. Up to eight processes can be shown in each display.) The /TOP qualifiers are conflicting qualifiers. Specify no more than one of them in a single request.

/ALL

Specifies that a table of all available statistics (current, average, minimum, and maximum) is to be included in the display and summary output. This qualifier is the default for the DECNET, FCP, IO, LOCK, PAGE, and POOL class-names.

/AVERAGE

Selects average statistics to be displayed in a bar graph for display and summary output.

/CPU
/NOCPU

Selects the processor-specific form of display and summary output for the MODES class (for VAX-11/782 attached processor configurations). This qualifier is valid only for the MODES class-name. If the attached processor is not active, the qualifier has no effect.

/CURRENT

Selects current statistics to be displayed in a bar graph for display and summary output. The /CURRENT qualifier is the default for the MODES and STATES class-names.

/MAXIMUM

Selects maximum statistics to be displayed in a bar graph for display and summary output.

/MINIMUM

Selects minimum statistics to be displayed in a bar graph for display and summary output.

/PERCENT
/NOPERCENT

Controls whether statistics are expressed as percent values in display and summary output. The /PERCENT qualifier is applicable only to the MODES and STATES class-names.

/TOPBIO

Used with the PROCESSES class-name to specify that a bar graph listing the top buffered I/O users be produced instead of the standard display and summary output. Values are expressed in units of buffered I/Os per second.

This qualifier can be used only with the PROCESSES class-name.

/TOPCPU

Used with the PROCESSES class-name to specify that a bar graph listing the top CPU time users be produced instead of the standard display and summary output. Values are expressed in units of clock ticks (10 milliseconds) per second.

This qualifier can only be used with the PROCESSES class-name.

/TOPDIO

Used with the PROCESSES class-name to specify that a bar graph listing the top direct I/O users be produced instead of the standard display and summary output. Values are expressed in units of direct I/Os per second.

This qualifier can only be used with the PROCESSES class-name.

/TOPFAULT

Used with the PROCESSES class-name to specify that a bar graph listing the top page faulting processes be produced instead of the standard display and summary output. Values are expressed in units of page faults per second.

This qualifier can only be used with the PROCESSES class-name.

12.3 OUTPUTS

The Monitor Utility can produce any combination of three forms of output for any single MONITOR request. The forms are display output, summary output, and recording file output.

12.3.1 Display Output

Display output consists of a series of terminal screen images. One screen image per requested class per requested viewing interval is produced. Any terminal supported by VAX/VMS with dimensions of at least 80 columns by 24 rows can be used. (You may have to issue the DCL command SET TERMINAL to set the proper dimensions.) Display output can also be routed to a file for subsequent printing.

The amount of time between screen displays is determined by the /VIEWING_TIME value. However, by pressing CTRL/W, you can generate a new display immediately following the current display. This feature is useful when the MONITOR display area has been overwritten by an operator message. You can also use CTRL/W in conjunction with a large /VIEWING_TIME value to generate display events on demand.

Section 12.4 provides full details of all the classes of display output you can obtain, with a sample of each.

12.3.1.1 Display Data - With the exception of the PROCESSES class, all displayable data items are rates or levels. Rates are shown in number of occurrences per second. A level is a value that indicates the size of the monitored data item.

MONITOR can display any of four different statistics for each data item, as follows:

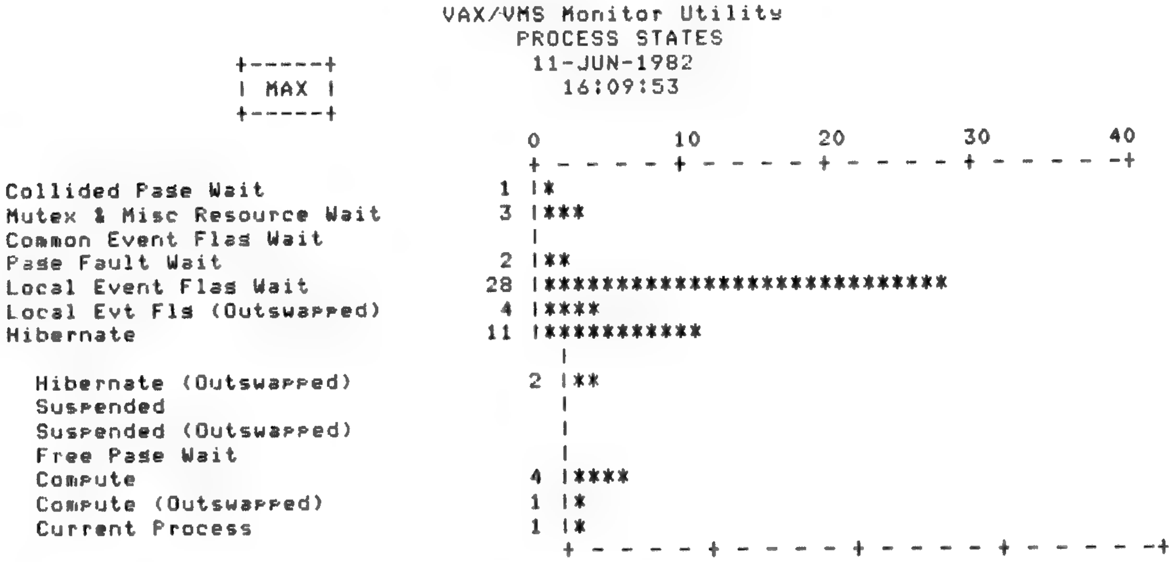
- Current rate or level
- Average rate or level
- Minimum rate or level
- Maximum rate or level

The last three statistics are measured since the beginning of the MONITOR request. The current statistic displays the most recently collected value for the rate or level. Any one or all four of the statistics can be requested. For certain classes, all the above statistics can be expressed as percentages. For a description of the display formats for the PROCESSES class, see Section 12.4.8.

12.3.1.2 Screen Formats - With the exception of the PROCESSES class, there are two basic screen formats used for displaying MONITOR class data: the single-statistic screen and the multiple-statistic screen. These screen formats are described in the next two sections.

Observe that there are two characteristics common to the screen formats. First, the date and time appearing in the heading of each screen refer to the time at which the displayed data was originally collected. Second, the bottom line of the display is used for status information pertaining to the current MONITOR request. If data collection is from a file of previously recorded monitor data, the word PLAYBACK appears at the left margin of the line. If, instead, the currently running system is being monitored, the word does not appear. If a summary file has been requested, the word SUMMARIZING appears in the middle of the line; otherwise, it does not appear. If creation of a recording file has been requested, the word RECORDING appears at the right margin of the line; otherwise, it does not appear.

12.3.1.2.1 **Single-Statistic Screen** - This bar-graph style screen is used whenever one statistic (current, average, minimum or maximum) is requested. The following sample screen exhibits the maximum statistic for the STATES class. For other classes and statistics, the screen format remains the same, with different heading and data item descriptions. If the display of percentages is requested, the percent symbol (%) appears in the title and next to the numbers along the top of the graph. All figures in this screen format are truncated to seven whole numbers (except percentages, which are truncated to three whole numbers). Figure 12-1 shows the single-statistic screen.



```
                                       VAX/VMS Monitor Utility
                                           PROCESS STATES
               +-----+                      11-JUN-1982
               | MAX |                        16:09:53
               +-----+
                                         0         10        20        30        40
                                         + - - - - + - - - - + - - - - + - - - - -+
Collided Page Wait                     1 |*
Mutex & Misc Resource Wait             3 |***
Common Event Flag Wait                   |
Page Fault Wait                        2 |**
Local Event Flag Wait                 28 |****************************
Local Evt Flg (Outswapped)             4 |****
Hibernate                             11 |***********
                                           |
  Hibernate (Outswapped)                 2 |**
  Suspended                                |
  Suspended (Outswapped)                   |
  Free Page Wait                           |
  Compute                                4 |****
  Compute (Outswapped)                   1 |*
  Current Process                        1 |*
                                           + - - - - + - - - - + - - - - + - - - - -+
```

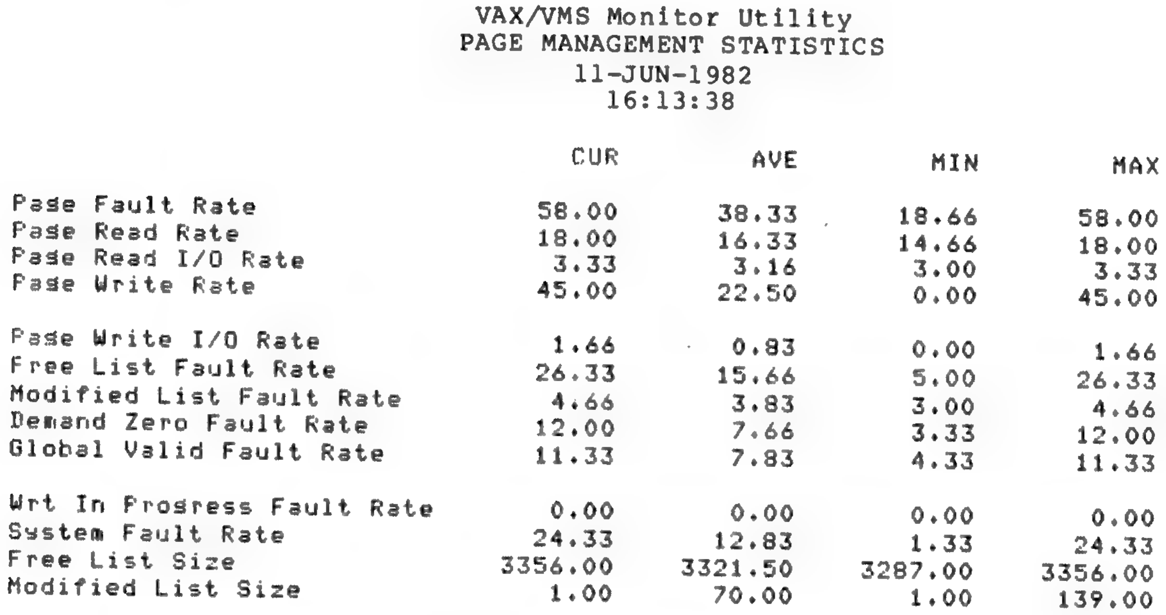
Figure 12-1: Sample Single-Statistic Screen

12.3.1.2.2 **Multiple-Statistic Screen** - This tabular-style screen is used whenever all four statistics are requested with the /ALL class-name qualifier. Figure 12-2 shows a multiple-statistic screen. The precision of the figures is: seven whole and two decimal places. For each class, the screen format remains the same, with different heading and data item descriptions.

If you request the display of percentages, as in the MODES class example in Figure 12-3, the percent sign (%) appears in the title and the headings, and the figures consist of three whole and one decimal place.

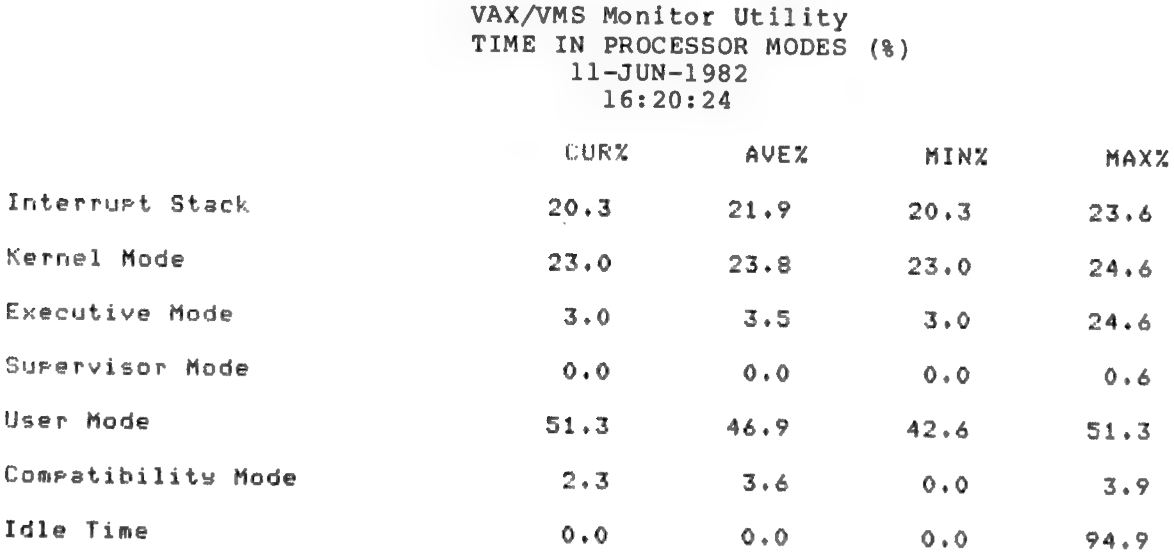
12.3.2 Summary Output

Summary output is an ASCII disk file consisting of one display screen image per requested class. The screen format for each class is based on the statistic requested. The only difference in format between a display screen and a summary screen image is that the word SUMMARY appears in the heading along with a beginning and ending time for the period covered by the summary. For all except the PROCESSES/TOP summaries (see Section 12.4.8), the data contained in the summaries is identical to that shown on the final display screen (if display output had also been requested). Since the summary file reflects the accumulation of data throughout the MONITOR request, the average, minimum, and maximum statistics are of particular interest. For the TOP summaries, the data represents the top users for the entire duration of the MONITOR request, subject to the following restriction. To be eligible for inclusion in the list of top users, a process must be present and swapped in at the beginning and end of the MONITOR request.



```
                              VAX/VMS Monitor Utility
                            PAGE MANAGEMENT STATISTICS
                                   11-JUN-1982
                                    16:13:38

                                   CUR        AVE        MIN        MAX

Page Fault Rate                   58.00      38.33      18.66      58.00
Page Read Rate                    18.00      16.33      14.66      18.00
Page Read I/O Rate                 3.33       3.16       3.00       3.33
Page Write Rate                   45.00      22.50       0.00      45.00

Page Write I/O Rate                1.66       0.83       0.00       1.66
Free List Fault Rate              26.33      15.66       5.00      26.33
Modified List Fault Rate           4.66       3.83       3.00       4.66
Demand Zero Fault Rate            12.00       7.66       3.33      12.00
Global Valid Fault Rate           11.33       7.83       4.33      11.33

Wrt In Progress Fault Rate         0.00       0.00       0.00       0.00
System Fault Rate                 24.33      12.83       1.33      24.33
Free List Size                  3356.00    3321.50    3287.00    3356.00
Modified List Size                 1.00      70.00       1.00     139.00
```

Figure 12-2: Sample Multiple-Statistic Screen



```
                              VAX/VMS Monitor Utility
                            TIME IN PROCESSOR MODES (%)
                                   11-JUN-1982
                                    16:20:24

                                   CUR%       AVE%       MIN%       MAX%

Interrupt Stack                    20.3       21.9       20.3       23.6

Kernel Mode                        23.0       23.8       23.0       24.6

Executive Mode                      3.0        3.5        3.0       24.6

Supervisor Mode                     0.0        0.0        0.0        0.6

User Mode                          51.3       46.9       42.6       51.3

Compatibility Mode                  2.3        3.6        0.0        3.9

Idle Time                           0.0        0.0        0.0       94.9
```

Figure 12-3: Sample Multiple-Statistic Screen (Data Expressed as Percentages)

12.3.3 Recording Files

A recording file is a VAX-11 RMS sequential disk file that is created when a MONITOR request includes the /RECORD qualifier. A record of binary performance data is written to this file once for each requested class per interval; the record contains a predefined set of data for each of the requested performance classes. The file is created when a MONITOR request is initiated and closed when the request terminates. The resulting file can be used as a source file by later requests to format and display the data on a terminal, to create a summary file, or to record a new recording file with different characteristics.

All data pertaining to the class is recorded. Note that this is true even if you are concurrently displaying only a single statistic.

A complete description of the MONITOR recording file formats appears in Appendix D.

12.4 MONITOR CLASSES

This section describes the classes of data you can monitor using the Monitor Utility. A sample display or summary of each class is provided with a brief description of the items in the class. The sections are presented in alphabetical order by class-name.

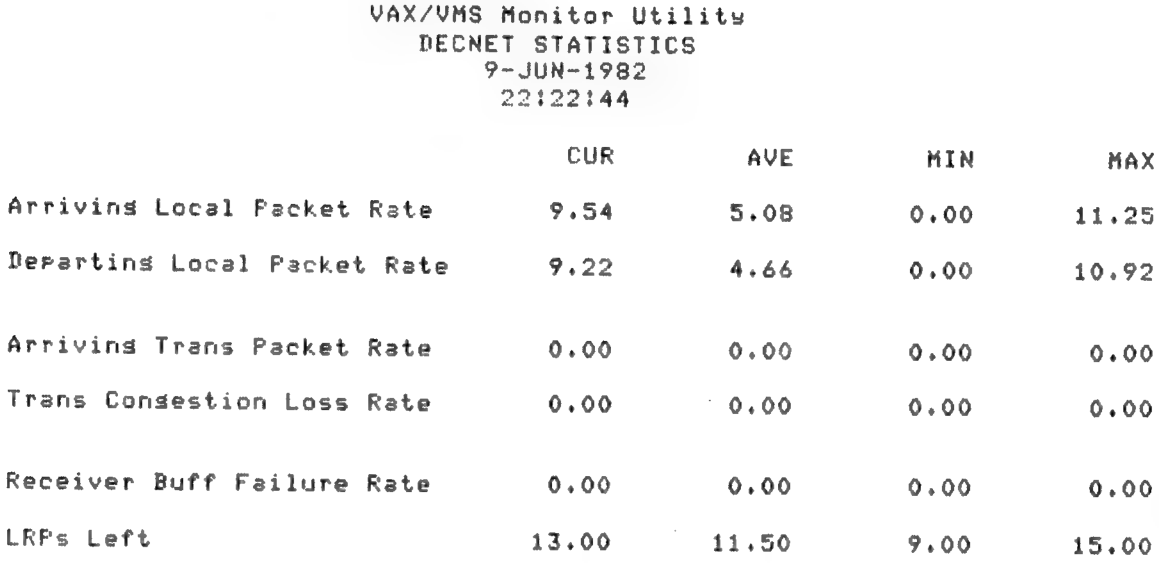
12.4.1 MONITOR DECNET

The MONITOR DECNET command initiates monitoring of the DECNET STATISTICS class, which includes information on DECnet-VAX network activity. It consists of the following data items:

- Arriving Local Packet Rate -- Rate at which local packets are being received
- Departing Local Packet Rate -- Rate at which local packets are being sent
- Arriving Trans Packet Rate -- Rate at which transit packets are arriving
- Trans Congestion Loss Rate -- Rate of transit congestion loss
- Receiver Buff Failure Rate -- Rate of receiver buffer failures
- LRPs Left -- Number of large request packets not in use

Figure 12-4 illustrates a DECNET STATISTICS display generated by the following command:

```
$ MONITOR DECNET
```



```
                          VAX/VMS Monitor Utility
                             DECNET STATISTICS
                                9-JUN-1982
                                 22:22:44

                                   CUR        AVE        MIN        MAX

Arriving Local Packet Rate        9.54       5.08       0.00      11.25

Departing Local Packet Rate       9.22       4.66       0.00      10.92


Arriving Trans Packet Rate        0.00       0.00       0.00       0.00

Trans Congestion Loss Rate        0.00       0.00       0.00       0.00


Receiver Buff Failure Rate        0.00       0.00       0.00       0.00

LRPs Left                        13.00      11.50       9.00      15.00
```

Figure 12-4: DECNET STATISTICS Display

This display shows, for example, that arriving and departing network packet rates (including control packets) are roughly equivalent, and that network activity is currently at a level higher than the average since monitoring began, but not at its highest point. Note also that the count of LRPs is displayed; LRPs are used by various components of VAX/VMS, but primarily by DECnet-VAX. A sufficient number of these large request packets must be available to ensure that at peak periods of network use they are not exhausted. If they become depleted, network performance may degrade. The number of packets preallocated at boot time is determined by the SYSGEN parameter LRPCNT.

12.4.2 MONITOR FCP

The MONITOR FCP command initiates monitoring of the FILE PRIMITIVE STATISTICS class, which includes information on all Files-11 ACPs on the system. It consists of the following data items, all of which are displayed as rates (that is, occurrences per second):

- FCP Call Rate -- Rate of QIO requests received by the system
- Allocation Rate -- Rate of calls that caused allocation of disk space
- Create Rate -- Rate at which new files were created
- Disk Read Rate -- Rate of read I/O operations from disk by the file system
- Disk Write Rate -- Rate of write I/O operations to disk by the file system

- Cache Hit Rate -- Rate at which requested blocks were located in the file system cache

- CPU Tick Rate -- Rate at which CPU time was used by the file system (in 10-millisecond ticks)

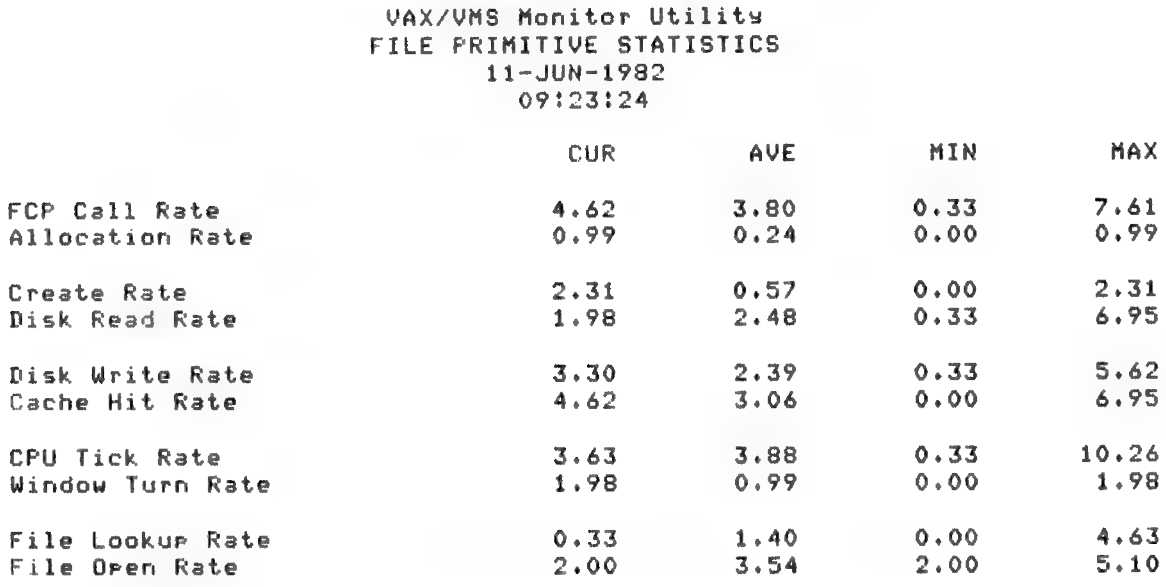
- Window Turn Rate -- Rate of file mapping window misses

- File Lookup Rate -- Rate of file name look-up operations in file directories

- File Open Rate -- Rate at which files were opened

Figure 12-5 illustrates a FILE PRIMITIVE STATISTICS display generated by the following command:

```
$ MONITOR FCP /INTERVAL=10
```



```
                                VAX/VMS Monitor Utility
                               FILE PRIMITIVE STATISTICS
                                      11-JUN-1982
                                        09:23:24

                                        CUR          AVE          MIN          MAX

FCP Call Rate                          4.62         3.80         0.33         7.61
Allocation Rate                        0.99         0.24         0.00         0.99

Create Rate                            2.31         0.57         0.00         2.31
Disk Read Rate                         1.98         2.48         0.33         6.95

Disk Write Rate                        3.30         2.39         0.33         5.62
Cache Hit Rate                         4.62         3.06         0.00         6.95

CPU Tick Rate                          3.63         3.88         0.33        10.26
Window Turn Rate                       1.98         0.99         0.00         1.98

File Lookup Rate                       0.33         1.40         0.00         4.63
File Open Rate                         2.00         3.54         2.00         5.10
```

Figure 12-5: FILE PRIMITIVE STATISTICS Display

This display shows, for example, that the rate of file opens during the last 10-second collection interval was 2.0 (for a total of 20). The average rate since the MONITOR command was issued is 3.54; the highest rate achieved during any 10-second interval is 5.10, and the lowest rate of 2.0 occurred during the last interval.

12.4.3 MONITOR IO

The MONITOR IO command initiates monitoring of the I/O SYSTEM STATISTICS class, which includes the following data items:

- Direct I/O Rate -- Rate of direct I/O (for example, disk and tape) operations

- Buffered I/O Rate -- Rate of buffered (for example, terminal and line printer) I/O operations

- Mailbox Write Rate -- Rate of write-to-mailbox requests received by the system

- Window Turn Rate -- Rate of file mapping window misses

- Log Name Translation Rate -- Rate of logical name translations

- File Open Rate -- Rate at which files were opened

- Page Fault Rate -- Rate of occurrence of page faults for all working sets

- Page Read Rate -- Rate of pages read from disk as a result of page faults

- Page Read I/O Rate -- Rate of read I/O operations from disk as a result of page faults

- Page Write Rate -- Rate of pages written to the paging file

- Page Write I/O Rate -- Rate of write I/O operations to the paging file

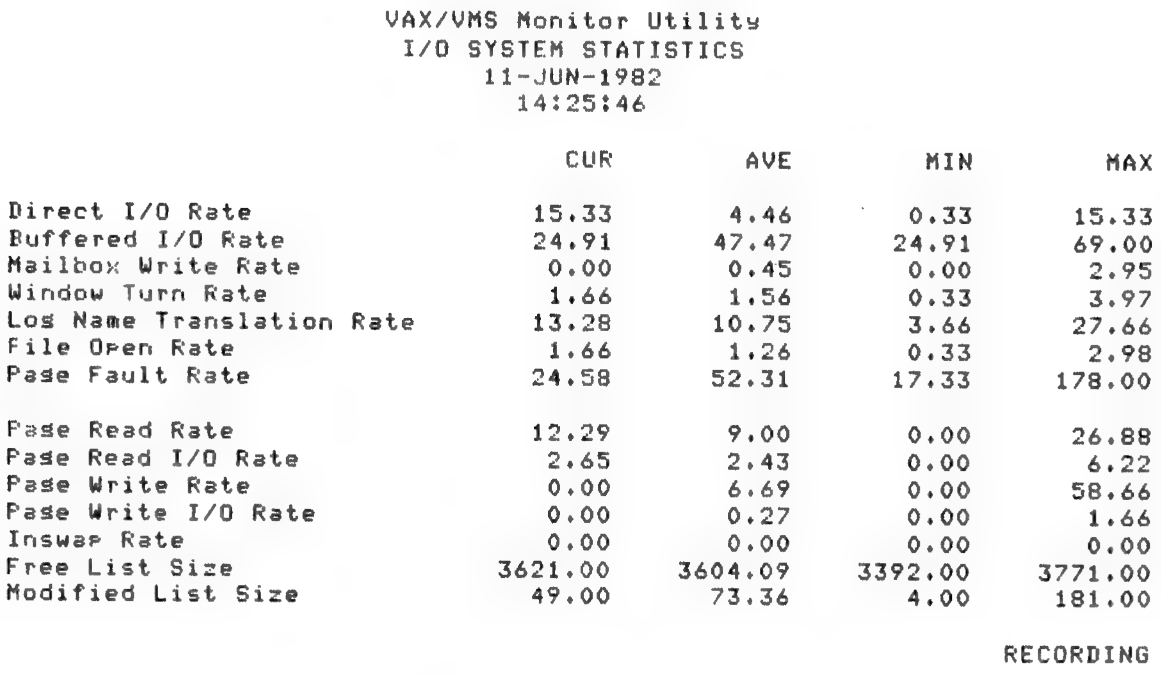
- Inswap Rate -- Rate at which working sets were read into memory from the swapping file

- Free List Size -- Number of pages on the free page list

- Modified List Size -- Number of pages on the modified page list

Figure 12-6 illustrates an I/O SYSTEM STATISTICS display generated by the following command:

```
$ MONITOR IO /RECORD
```



```
                         VAX/VMS Monitor Utility
                          I/O SYSTEM STATISTICS
                               11-JUN-1982
                                 14:25:46

                                CUR         AVE         MIN         MAX

Direct I/O Rate               15.33        4.46        0.33       15.33
Buffered I/O Rate             24.91       47.47       24.91       69.00
Mailbox Write Rate             0.00        0.45        0.00        2.95
Window Turn Rate               1.66        1.56        0.33        3.97
Log Name Translation Rate     13.28       10.75        3.66       27.66
File Open Rate                 1.66        1.26        0.33        2.98
Page Fault Rate               24.58       52.31       17.33      178.00

Page Read Rate                12.29        9.00        0.00       26.88
Page Read I/O Rate             2.65        2.43        0.00        6.22
Page Write Rate                0.00        6.69        0.00       58.66
Page Write I/O Rate            0.00        0.27        0.00        1.66
Inswap Rate                    0.00        0.00        0.00        0.00
Free List Size              3621.00     3604.09     3392.00     3771.00
Modified List Size            49.00       73.36        4.00      181.00

                                                              RECORDING
```

Figure 12-6: I/O SYSTEM STATISTICS Display

This display shows, for example, that the direct I/O rate is currently at its highest level since the MONITOR command was issued, and is significantly higher than the average rate. Termination of this command by CTRL/C and issuance of a MONITOR PROCESSES/TOPDIO command would show the top users of direct I/Os. Note that if I/O monitoring is begun at a later time, a new MONITOR request is defined. That is, it is not a continuation of the original request; the average, minimum and maximum statistics are reinitialized. However, since the original request selected recording, that data can be played back for redisplay or summarization.

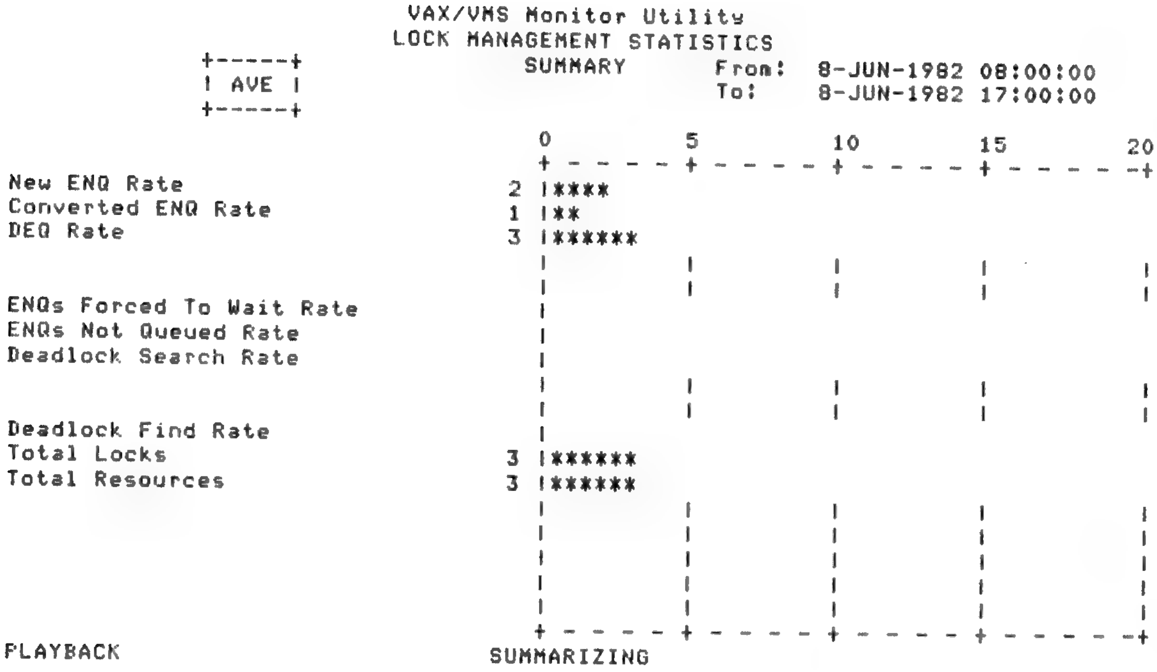
12.4.4 MONITOR LOCK

The MONITOR LOCK command initiates monitoring of the LOCK MANAGEMENT STATISTICS class, which includes the following data items:

- New ENQ Rate -- Rate of new lock (ENQ) requests (as opposed to conversions)
- Converted ENQ Rate-- Rate of lock (ENQ) conversion requests
- DEQ Rate -- Rate of unlock (DEQ) requests
- ENQs Forced To Wait Rate -- Rate of occurrence of locks that could not be granted immediately, thus having to wait
- ENQs Not Queued Rate -- Rate of occurrence of locks that could not be granted immediately but requested not to be queued, and thus received an error status instead
- Deadlock Search Rate -- Rate at which a deadlock search was performed
- Deadlock Find Rate -- Rate at which a deadlock was found
- Total Locks -- Total number of locks in the system
- Total Resources -- Total number of resources in the system

Figure 12-7 illustrates how to generate and display a LOCK MANAGEMENT STATISTICS summary using the following commands:

```
$ MONITOR LOCK/AVERAGE /INPUT=LOCKSTATS.DAT/SUMMARY/NODISPLAY
$ TYPE MONITOR.SUM
```



```
                                   VAX/VMS Monitor Utility
                                  LOCK MANAGEMENT STATISTICS
              +-----+                    SUMMARY        From:   8-JUN-1982 08:00:00
              ! AVE !                                   To:     8-JUN-1982 17:00:00
              +-----+
                                          0         5         10        15        20
                                          + - - - - + - - - - + - - - - + - - - - -+
New ENQ Rate                            2 |****
Converted ENQ Rate                      1 |**
DEQ Rate                                3 |******
                                          |         |         |         |          |
                                          |         |         |         |          |
ENQs Forced To Wait Rate                  |
ENQs Not Queued Rate                      |
Deadlock Search Rate                      |
                                          |         |         |         |          |
                                          |         |         |         |          |
Deadlock Find Rate                        |
Total Locks                             3 |******
Total Resources                         3 |******
                                          |         |         |         |          |
                                          |         |         |         |          |
                                          |         |         |         |          |
                                          |         |         |         |          |
                                          |         |         |         |          |
                                          + - - - - + - - - - + - - - - + - - - - -+
PLAYBACK                                SUMMARIZING
```

Figure 12-7: LOCK MANAGEMENT STATISTICS Summary

This display shows the average use of the lock management subsystem during a typical workday, based on data that was previously recorded.

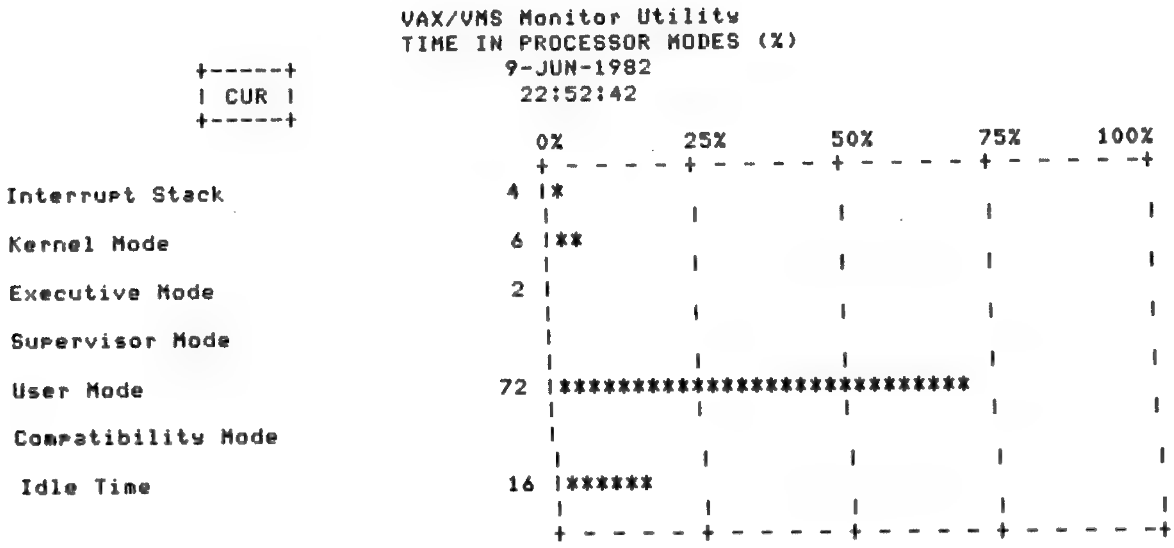
12.4.5 MONITOR MODES

The MONITOR MODES command initiates monitoring of the TIME IN PROCESSOR MODES class, which includes a data item for each mode of processor operation. The following data items can be displayed as percentages of all processor (CPU) time, or as rates of clock ticks (10 millisecond units) per second:

- Interrupt Stack -- Time spent on the interrupt stack
- Kernel Mode -- Time spent in kernel mode, but not on interrupt stack
- Executive Mode -- Time spent in executive mode
- Supervisor Mode -- Time spent in supervisor mode
- User Mode -- Time spent in user mode executing VAX-11 instructions
- Compatibility Mode -- Time spent executing compatibility-mode instructions
- Idle Time -- Time spent executing the NULL process

Figure 12-8 illustrates a TIME IN PROCESSOR MODES display generated by the following command:

```
     $ MONITOR MODES /PERCENT
```



```
                                  VAX/VMS Monitor Utility
                                TIME IN PROCESSOR MODES (%)
                +-----+                 9-JUN-1982
                | CUR |                  22:52:42
                +-----+
                                      0%        25%       50%       75%      100%
                                      + - - - - + - - - - + - - - - + - - - - -+
Interrupt Stack                     4 |*
                                      |         |         |         |          |
Kernel Mode                         6 |**
                                      |         |         |         |          |
Executive Mode                      2 |
                                      |         |         |         |          |
Supervisor Mode                       |
                                      |         |         |         |          |
User Mode                          72 |****************************
                                      |         |         |         |          |
Compatibility Mode                    |
                                      |         |         |         |          |
 Idle Time                         16 |******
                                      |         |         |         |          |
                                      + - - - - + - - - - + - - - - + - - - - -+
```

Figure 12-8: TIME IN PROCESSOR MODES Display

This display shows, for example, that over the last collection interval, the processor spent 72% of its time executing user code, 8% executing system code to service user requests in executive and kernel modes, 4% processing interrupts on the interrupt stack, and it was idle 16% of the time. Time spent executing VAX-11 RMS code is included in executive mode time. Time spent executing DCL code is included in supervisor mode time. The majority of interrupt stack time is devoted to processing buffered I/O requests.

If you omit the /PERCENT qualifier or specify /NOPERCENT, MONITOR displays mode times as rates of clock ticks per second, where a clock tick is 10 milliseconds.

For a VAX-11/782 attached processor configuration, the MODES display consists of 14 items representing the seven modes for each of the two processors. If you specify /NOCPU on such a system, only the seven modes are displayed, where each mode includes the total time for both processors. On a single processor system, the /CPU qualifier has no effect.

12.4.6 MONITOR PAGE

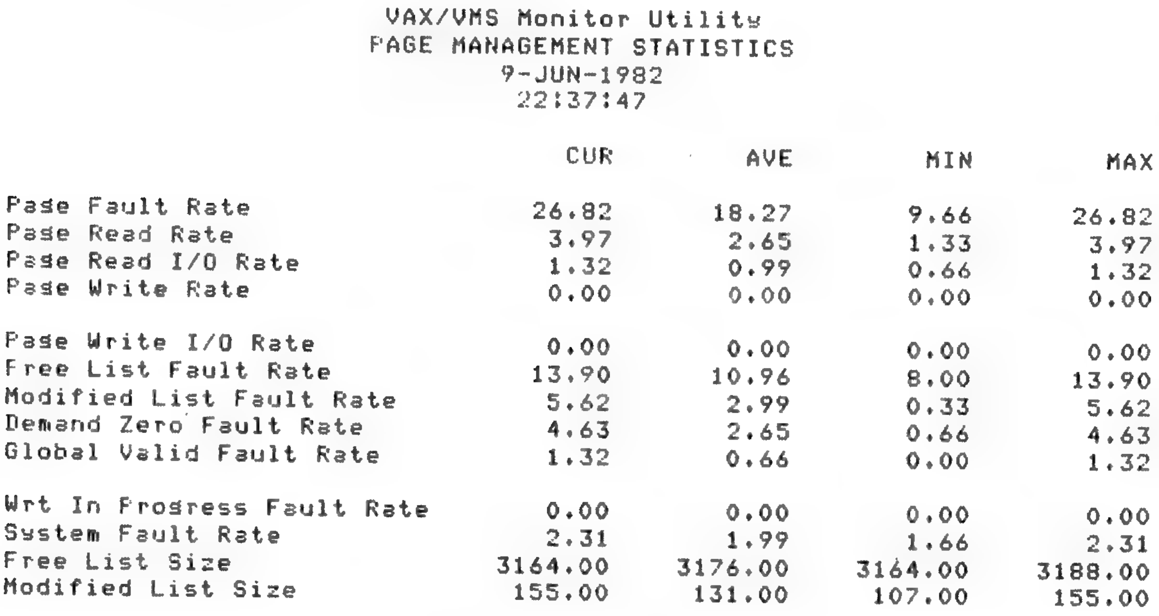
The MONITOR PAGE command initiates monitoring of the PAGE MANAGEMENT STATISTICS class, which includes the following data items:

- Page Fault Rate -- Rate of page faults for all working sets
- Page Read Rate -- Rate of pages read from disk as a result of page faults

- Page Read I/O Rate -- Rate of read I/O operations from disk as a result of page faults
- Page Write Rate -- Rate at which pages were written to the page file
- Page Write I/O Rate -- Rate of write I/O operations to the paging file
- Free List Fault Rate -- Rate at which pages were read from the free page list as a result of page faults
- Modified List Fault Rate -- Rate of pages read from the modified page list as a result of page faults
- Demand Zero Fault Rate -- Rate at which zero-filled pages were allocated as a result of page faults
- Global Valid Fault Rate -- Rate of page faults for pages that are not in the process's working set but are in physical memory and are indicated as valid pages in the system-wide global page tables
- Wrt in Progress Fault Rate -- Rate of pages read that were in the process of being written back to disk, when faulted
- System Fault Rate -- Rate of page faults for pages in system space
- Free List Size -- Number of pages on the free page list
- Modified List Size -- Number of pages on the modified page list

Figure 12-9 illustrates a PAGE MANAGEMENT STATISTICS display generated by the following command:

```
$ MONITOR PAGE
```



```
                         VAX/VMS Monitor Utility
                       PAGE MANAGEMENT STATISTICS
                              9-JUN-1982
                               22:37:47

                                   CUR        AVE        MIN        MAX

Page Fault Rate                  26.82      18.27       9.66      26.82
Page Read Rate                    3.97       2.65       1.33       3.97
Page Read I/O Rate                1.32       0.99       0.66       1.32
Page Write Rate                   0.00       0.00       0.00       0.00

Page Write I/O Rate               0.00       0.00       0.00       0.00
Free List Fault Rate             13.90      10.96       8.00      13.90
Modified List Fault Rate          5.62       2.99       0.33       5.62
Demand Zero Fault Rate            4.63       2.65       0.66       4.63
Global Valid Fault Rate           1.32       0.66       0.00       1.32

Wrt In Progress Fault Rate        0.00       0.00       0.00       0.00
System Fault Rate                 2.31       1.99       1.66       2.31
Free List Size                 3164.00    3176.00    3164.00    3188.00
Modified List Size              155.00     131.00     107.00     155.00
```

Figure 12-9: PAGE MANAGEMENT STATISTICS Display

This display shows, for example, that, the current rate of pages read per read I/O operation is approximately 3 per second (Page Read Rate divided by Page Read I/O Rate). Note that while the page fault rate is currently at the highest point of the monitoring session, the vast majority of the pages are faulted from memory, not from disk.

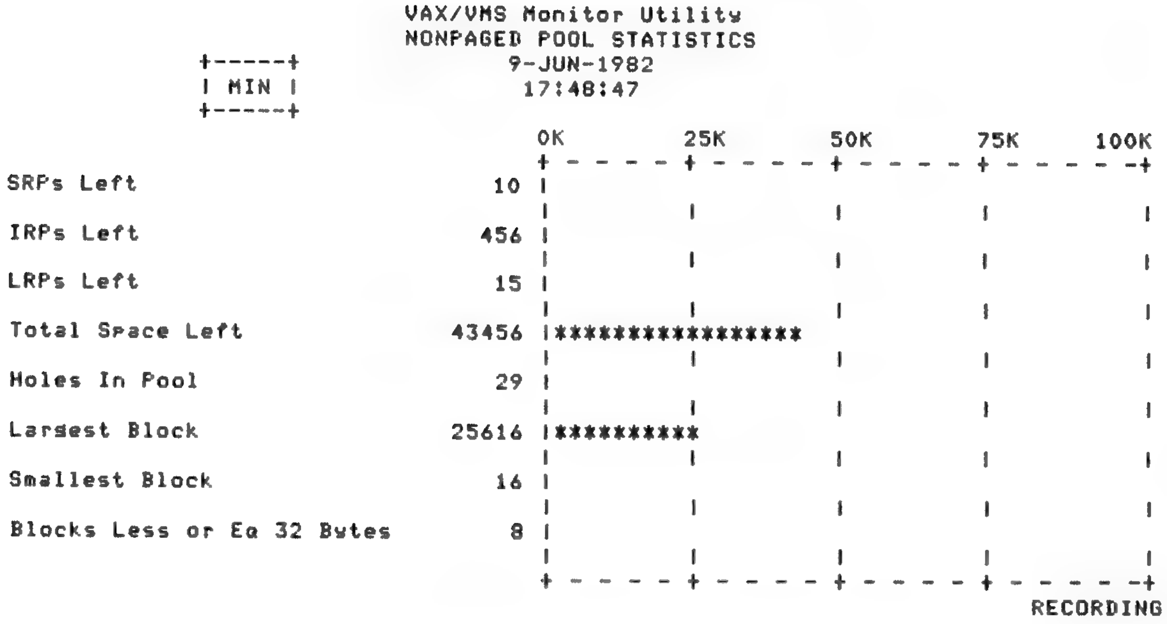
12.4.7 MONITOR POOL

The MONITOR POOL command initiates monitoring of the NONPAGED POOL STATISTICS class, which measures space allocations in the nonpaged dynamic pool. It includes the following data items:

- SRPs Left -- Number of small request packets available in the SRP queue
- IRPs Left -- Number of intermediate request packets available in the IRP queue
- LRPs Left -- Number of large request packets available in the LRP queue
- Total Space Left -- Total unused bytes in the dynamically allocated portion of the pool
- Holes in Pool -- Unused blocks of contiguous space in the dynamically allocated portion of the pool
- Largest Block -- Size in bytes of the largest block of unused space in the dynamically allocated portion of the pool
- Smallest Block -- Size in bytes of the smallest block of unused space in the dynamically allocated portion of the pool
- Blocks Less or Eq 32 Bytes -- Blocks less than or equal to 32 bytes in size in the dynamically allocated portion of the pool

Figure 12-10 illustrates a NONPAGED POOL STATISTICS display generated by the following command:

```
$ MONITOR POOL/MINIMUM /RECORD
```



```
                                   VAX/VMS Monitor Utility
                                   NONPAGED POOL STATISTICS
                 +-----+                  9-JUN-1982
                 | MIN |                   17:48:47
                 +-----+
                                           0K        25K       50K       75K      100K
                                           + - - - - + - - - - + - - - - + - - - - -+
SRPs Left                             10 |
                                           |         |         |         |          |
IRPs Left                            456 |
                                           |         |         |         |          |
LRPs Left                             15 |
                                           |         |         |         |          |
Total Space Left                   43456 |*****************
                                           |         |         |         |          |
Holes In Pool                         29 |
                                           |         |         |         |          |
Largest Block                      25616 |**********
                                           |         |         |         |          |
Smallest Block                        16 |
                                           |         |         |         |          |
Blocks Less or Eq 32 Bytes             8 |
                                           |         |         |         |          |
                                           + - - - - + - - - - + - - - - + - - - - -+
                                                                              RECORDING
```

Figure 12-10: NONPAGED POOL STATISTICS Display

In this case, only the minimum statistic is being displayed, but data is being recorded that could later be used to redisplay all statistics. Of particular interest in this display are the SRPs, IRPs, and LRPs Left items. If queue sizes for these preallocated packets drop to 0 at any time, extra overhead is incurred by the subsystems requiring these packets. Initial queue sizes of these items are determined by system parameters that can be adjusted at boot time.

12.4.8 MONITOR PROCESSES

The MONITOR PROCESSES command initiates monitoring of the PROCESSES class, which displays information on all processes in the system. As illustrated below, the PROCESSES display (and summary) formats are different from those of all other classes.

The PROCESSES display provides the following information:

- PID -- Process identification as assigned by the system, in hexadecimal
- UIC -- User identification code, in octal
- STATE -- Process's scheduler state
- PRI -- Current (as opposed to base) priority of the process
- NAME -- Process name
- SIZE -- Number of shareable pages and total number of pages in use by the process

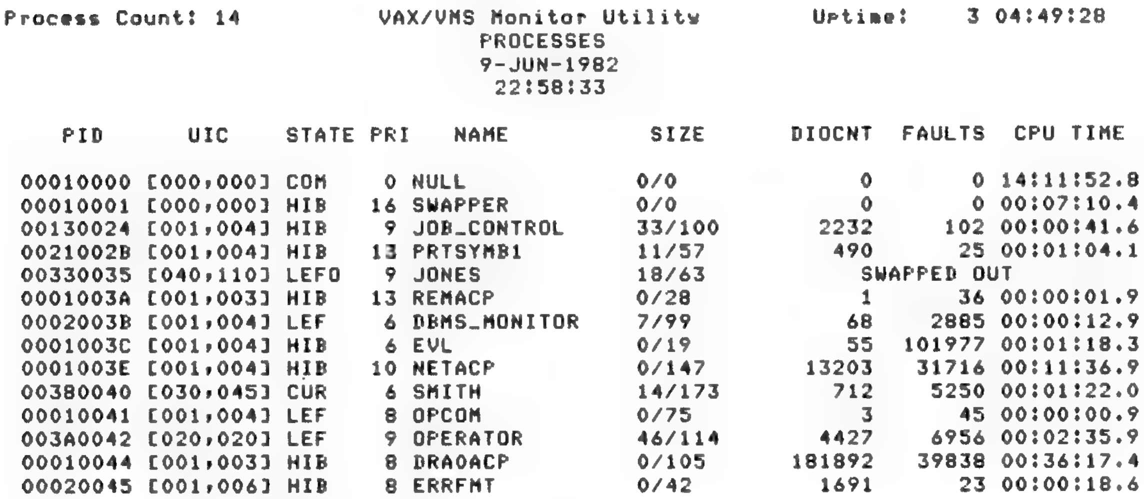
- DIOCNT -- Cumulative direct I/O operations performed by the process since its creation; not displayed if the process is swapped out
- FAULTS -- Cumulative page faults since the process was created; not displayed if process is swapped out
- CPU TIME -- Cumulative CPU time used by the process since its creation, in the format hours:minutes:seconds; not displayed if process is swapped out

The top corners of the display contain the number of processes in the system and the time in days, hours, minutes, and seconds since the system was last booted. Processes that are swapped out are so noted.

If more processes are in the system than can be displayed on the terminal screen at once, the full display occurs in screenfuls, presented one at a time where each display is separated by a two-second wait.

Figure 12-11 illustrates a PROCESSES display generated by the following command:

```
$ MONITOR PROCESSES /INPUT=PROCS.DAT
```



```
Process Count: 14          VAX/VMS Monitor Utility         Uptime:     3 04:49:28
                                  PROCESSES
                                 9-JUN-1982
                                  22:58:33

     PID       UIC     STATE PRI   NAME            SIZE       DIOCNT  FAULTS  CPU TIME

 00010000 [000,000] COM    0 NULL             0/0              0       0 14:11:52.8
 00010001 [000,000] HIB   16 SWAPPER          0/0              0       0 00:07:10.4
 00130024 [001,004] HIB    9 JOB_CONTROL      33/100        2232     102 00:00:41.6
 0021002B [001,004] HIB   13 PRTSYMB1         11/57          490      25 00:01:04.1
 00330035 [040,110] LEFO   9 JONES            18/63            SWAPPED OUT
 0001003A [001,003] HIB   13 REMACP           0/28             1      36 00:00:01.9
 0002003B [001,004] LEF    6 DBMS_MONITOR     7/99            68    2885 00:00:12.9
 0001003C [001,004] HIB    6 EVL              0/19            55  101977 00:01:18.3
 0001003E [001,004] HIB   10 NETACP           0/147        13203   31716 00:11:36.9
 00380040 [030,045] CUR    6 SMITH            14/173         712    5250 00:01:22.0
 00010041 [001,004] LEF    8 OPCOM            0/75             3      45 00:00:00.9
 003A0042 [020,020] LEF    9 OPERATOR         46/114        4427    6956 00:02:35.9
 00010044 [001,003] HIB    8 DRA0ACP          0/105       181892   39838 00:36:17.4
 00020045 [001,006] HIB    8 ERRFMT           0/42          1691      23 00:00:18.6
```

PLAYBACK

Figure 12-11: PROCESSES Display

One line is displayed for each process in the system. Note that this display shows current values only -- average, minimum, and maximum statistics are not available. Also note that for swapped-out processes, the words SWAPPED OUT replace the three rightmost items, since those items are not available for swapped-out processes. Since this example is a playback request, the system uptime displayed is that of the system at the time the MONITOR data was recorded.

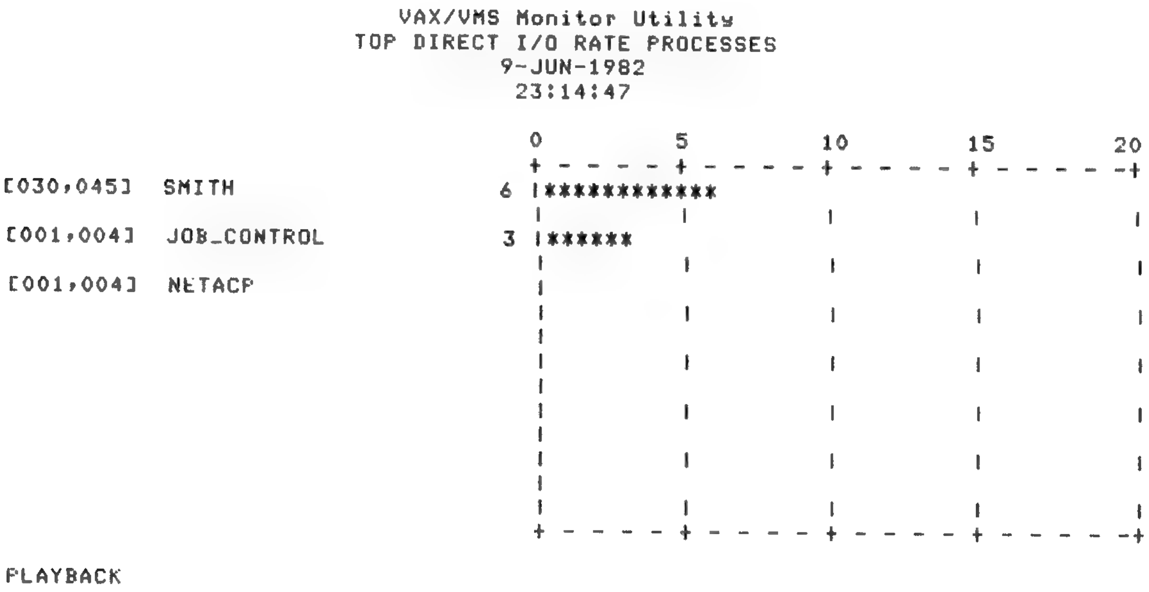
Nondisplayable characters in process names are represented by periods.

Figure 12-12 illustrates a PROCESSES display generated by the following command:

```
     $ MONITOR PROCESSES/TOPDIO /INPUT=PROCS.DAT
```

Using the same input file as above, this display shows that the process SMITH, with a rate of 6 per second, was the top consumer of direct I/Os during the most recent interval between displays.

As with the other bar graph displays, figures in the displays of top users are truncated to the nearest whole number. Up to eight processes with nonzero values are displayed. To be eligible for inclusion in the list of top users, a process must be present and swapped in at the beginning and end of the display interval. This eligibility requirement also applies to the beginning and ending of the entire period covered by a summary.



```
                                VAX/VMS Monitor Utility
                             TOP DIRECT I/O RATE PROCESSES
                                      9-JUN-1982
                                       23:14:47

                                         0         5         10        15        20
                                         + - - - - + - - - - + - - - - + - - - - -+
[030,045]  SMITH                       6 |************
                                         |         |         |         |          |
[001,004]  JOB_CONTROL                 3 |******
                                         |         |         |         |          |
[001,004]  NETACP                        |
                                         |         |         |         |          |
                                         |
                                         |         |         |         |          |
                                         |
                                         |         |         |         |          |
                                         |
                                         |         |         |         |          |
                                         |
                                         |         |         |         |          |
                                         + - - - - + - - - - + - - - - + - - - - -+

PLAYBACK
```

Figure 12-12: TOP DIRECT I/O RATE PROCESSES Display

Three other displays of top users are available. The /TOPBIO, /TOPFAULT, and /TOPCPU qualifiers produce displays of processes that are the top consumers of buffered I/Os, page faults, and processor time, respectively. All values are expressed as rates of occurrences per second, except those resulting from a /TOPCPU request, which are expressed as the number of 10-millisecond clock ticks per second. Only one of the displays of top users or the regular PROCESSES display can be selected in a single MONITOR request.

12.4.9 MONITOR STATES

The MONITOR STATES command initiates monitoring of the PROCESS STATES class, which shows the number of processes in each of the 14 scheduler states, as follows:

- Collided Page Wait -- Waiting for a faulted page in transition
- Mutex & Misc Resource Wait -- Waiting for the availability of a mutual exclusion semaphore or a dynamic resource
- Common Event Flag Wait -- Waiting for some combination of event flags to be set in a common event block
- Page Fault Wait -- Waiting for a page to be read as a result of a page fault; resident processes
- Local Event Flag Wait -- Waiting for one or more local event flags to be posted; resident processes
- Local Evt Flg (Outswapped) -- Waiting for one or more local event flags to be posted; outswapped processes
- Hibernate -- Hibernating, or process has executed a hibernate request; resident processes
- Hibernate (Outswapped) -- Hibernating, or process has executed a hibernate request; outswapped processes
- Suspended -- Process has executed a suspend request; resident processes
- Suspended (Outswapped) -- Process has executed a suspend request; outswapped processes
- Free Page Wait -- Waiting for a free page of memory
- Compute -- Ready to use the processor; resident processes
- Compute (Outswapped) -- Ready to use the processor; outswapped processes
- Current Process -- Using the processor

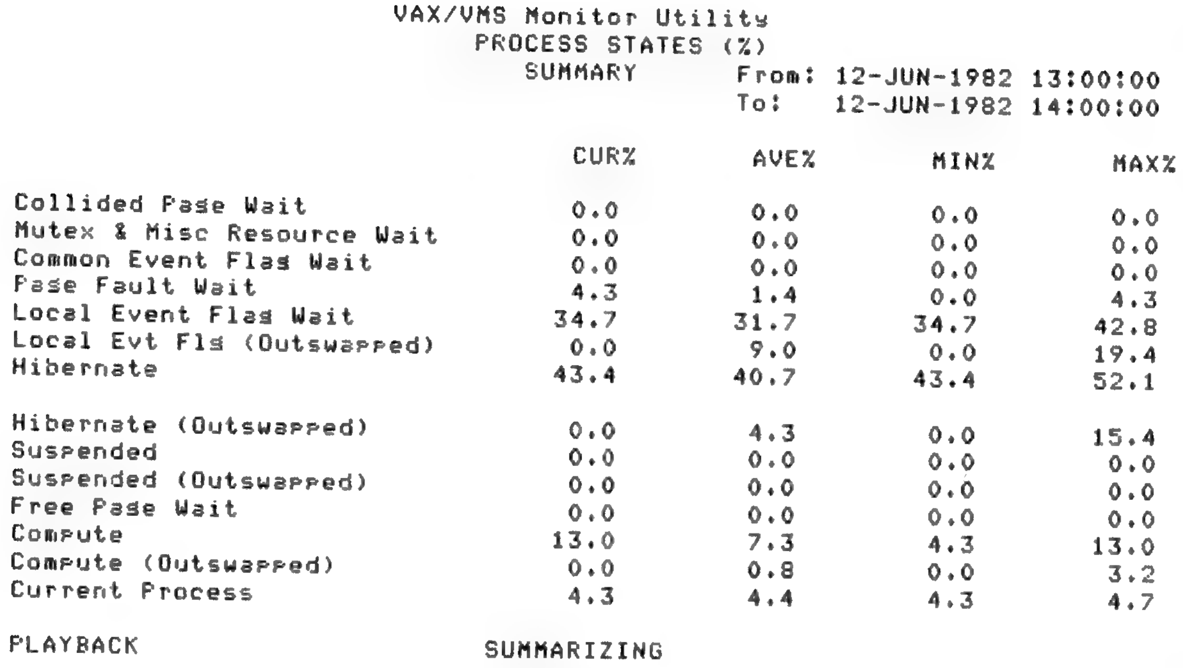
The data items can also be displayed as percentages of all processes.

Note that the Current Process is always the process running MONITOR, since MONITOR is running when each measurement is made. Local Event Flag Wait (Outswapped) processes normally belong to interactive users who have been prompted but have not responded, although they might be processes waiting for disk I/O on a crowded system. A state of Compute (Outswapped) for any process indicates a very crowded system.

Since, for performance reasons, the scanning of system data structures containing process state information is not synchronized with operating system use of those structures, there is a low probability that certain anomalous state indications will be displayed.

Figure 12-13 illustrates how to generate and display a PROCESS STATES summary file by using the following commands:

```
$ MONITOR STATES/PERCENT/ALL /INPUT/SUMMARY/NODISPLAY -
$_/BEGINNING=12-JUN-1982:13:00 -
$_/ENDING=12-JUN-1982:14:00
$_TYPE MONITOR.SUM
```



```
                             VAX/VMS Monitor Utility
                                PROCESS STATES (%)
                                  SUMMARY       From: 12-JUN-1982 13:00:00
                                                To:   12-JUN-1982 14:00:00

                                    CUR%        AVE%        MIN%        MAX%

Collided Page Wait                   0.0         0.0         0.0         0.0
Mutex & Misc Resource Wait           0.0         0.0         0.0         0.0
Common Event Flag Wait               0.0         0.0         0.0         0.0
Page Fault Wait                      4.3         1.4         0.0         4.3
Local Event Flag Wait               34.7        31.7        34.7        42.8
Local Evt Flg (Outswapped)           0.0         9.0         0.0        19.4
Hibernate                           43.4        40.7        43.4        52.1

Hibernate (Outswapped)               0.0         4.3         0.0        15.4
Suspended                            0.0         0.0         0.0         0.0
Suspended (Outswapped)               0.0         0.0         0.0         0.0
Free Page Wait                       0.0         0.0         0.0         0.0
Compute                             13.0         7.3         4.3        13.0
Compute (Outswapped)                 0.0         0.8         0.0         3.2
Current Process                      4.3         4.4         4.3         4.7

PLAYBACK                        SUMMARIZING
```

Figure 12-13: PROCESS STATES Summary

This summary shows, for example, that at some point during the requested period, the percentage of processes swapped out reached 38%. Note that the summary was requested for data covering only the hour between 1 P.M. and 2 P.M., although the input file could have contained data covering a longer period.

12.5 BASIC MODES OF OPERATION

This section describes some of the ways you can use MONITOR at your site.

12.5.1 Live Display Monitoring

Use this mode when you want to examine the activity of the running system, either on a routine basis, or as part of an installation checkout, tuning, or trouble-shooting exercise. No historical record of output is kept.

The following examples illustrate the live display monitoring mode of operation.

```
$ MONITOR PAGE
```

This command produces a display of page management statistics. Since no command or class-name qualifiers have been named, MONITOR applies the following defaults to the display:

```
/ALL /INTERVAL=3/VIEWING_TIME=3
/INPUT=the running system
/OUTPUT=current SYS$OUTPUT device
```

By default, monitoring begins when the command is issued and ends when the user issues a CTRL/C.

The next command displays a bar graph showing the eight processes that were the top consumers of CPU time during the period between displays. It also displays the amount of CPU time each of these processes used.

```
     $ MONITOR PROCESSES/TOPCPU
```

MONITOR display output can be routed to any supported terminal device, or to a disk file. The next command writes its display of nonpaged pool statistics to the file POOL.LOG. This file could then be printed out on a hard-copy device.

```
     $ MONITOR POOL /DISPLAY=POOL.LOG
```

It may be convenient to establish DCL symbols for frequently used combinations of class-names, as in the example below.

```
     $ ALL_CLASSES :== -
     $_"DECNET+FCP+IO+LOCK+MODES+PAGE+POOL+PROCESSES+STATES"
     $ MONITOR 'ALL_CLASSES' /INTERVAL=20/VIEWING_TIME=2
```

The MONITOR command collects all classes of data every 20 seconds. Every 2 seconds, the most recently collected data for one of the classes is displayed. The ordering of the classes for display is predetermined by MONITOR.

12.5.2 Live Recording

Use live recording whenever you need to capture MONITOR data for future use. Possible uses include the following:

- Installation checkout, tuning, trouble-shooting -- that is, all the uses listed above for live display monitoring. Choose recording over display whenever you would like to capture more classes than you could reasonably watch at a terminal, whenever a terminal is not available, or whenever you need to gather data about the system but cannot devote your time to the terminal until later.

- Routine performance data gathering for long-term analysis. MONITOR data can be recorded on a routine basis and summarized to gather data about system resource utilization over long periods of time.

The following example illustrates the live recording mode of operation.

```
     $ MONITOR MODES+STATES /NODISPLAY/RECORD
```

This command records data on the time spent in each of the processor modes and on the number of processes in each of the scheduler states, but it does not display this information.

12.5.3 Concurrent Display and Recording

The first two modes can be combined whenever you want to retain performance data and watch as it is being collected. The following example illustrates the concurrent display and recording mode of operation:

```
     $ MONITOR FCP/CURRENT,POOL/MINIMUM,FCP/AVERAGE /RECORD
```

This command collects and records file system ACP data and nonpaged dynamic pool data every three seconds. It also displays, in bar-graph form, average FCP statistics and minimum POOL statistics. The display alternates between the two graphs every three seconds. Note that the FCP class was requested twice; while this is a misuse of the MONITOR command, it is not flagged as an error. Since MONITOR allows only one statistic qualifier per class-name, only the latest specification (/AVERAGE) is used. Current statistics can be obtained in a subsequent playback request.

12.5.4 Playback

Use playback of a recording file to obtain terminal output and/or summary reports of all or just a subset of collected data. Data can be subsetted by class or time segment. For example, if several classes of data have been collected for an entire day, you can examine or summarize the data on any one or more of the classes during any time period in that day. You can also display or summarize data with a different interval than that at which it was recorded. The actual amount of time between displays of screen images is controlled with the /VIEWING_TIME qualifier.

The following examples illustrate the playback mode of operation.

```
     $ MONITOR IO /RECORD/INTERVAL=5
                   .
                   .
                   .
     $ MONITOR IO /INPUT
```

These commands produce system I/O statistics. The first command gathers and displays data every five seconds, beginning when the command is issued and ending when the user issues a CTRL/C. In addition, it records binary data in the default output file MONITOR.DAT. The second command plays back the I/O statistics display, using the data in MONITOR.DAT for input. The default viewing time for the playback is three seconds, but each screen display represents five seconds of monitored I/O statistics.

The next sequence of commands illustrates the recording of data with a relatively small interval and playback with a relatively large interval. This is useful for producing average, minimum, and maximum statistics that cover a wide range of time, but have greater precision than they would have if they had been gathered over the larger interval.

```
     $ MONITOR POOL /RECORD/NODISPLAY -
     $_/BEGINNING=08:00:00 -
     $_/ENDING=16:00:00 -
     $_/INTERVAL=120
                   .
                   .
                   .
     $ MONITOR POOL /INPUT/DISPLAY=HOURLY.LOG -
     $_/INTERVAL=3600
```

The first command above records data on space allocation in the nonpaged dynamic pool for the indicated eight-hour period, using an interval of two minutes. The second plays the data back with an interval of one hour, storing display output in the file HOURLY.LOG. This file can then be typed or printed to show the cumulative pool utilization at each hour throughout the eight-hour period.

The next command uses the recording file created in the previous example to produce a one-page summary report file showing the average statistics for the indicated eight-hour period.

```
$ MONITOR POOL/AVERAGE /INPUT/NODISPLAY/SUMMARY=DAILY.LOG
```

The summary report has the same format as a screen display, which in this case is a bar graph.

12.5.5 Rerecording

Rerecording is a combination of playback and recording. It can be useful for data reduction of recording files. When playing back an existing recording file, all MONITOR options are available to you; thus, you can choose to record a subset of the recorded classes and/or a subset of the recorded time segment and even a larger interval value. All these techniques produce a new, smaller, recording file, at the expense of some of the recorded data. A larger interval value reduces the volume of the collected data, so that displays and summary output produced from the newer recorded file will be less precise. Note that average rate values will not be affected in this case, but average level values will be less precise (since the sample size is reduced), as will maximum and minimum values.

The following example illustrates the rerecording mode of operation:

```
$ SUBMIT MONREC.COM
```

where MONREC.COM is:

```
$ MONITOR DECNET,LOCK /NODISPLAY/RECORD/INTERVAL=60 -
                      /BEGINNING=8:00/ENDING=16:00

$ MONITOR DECNET /INPUT/NODISPLAY/RECORD
```

The first command runs in a batch job, recording DECnet-VAX and lock management data once per minute, between the hours of 8 A.M. and 4 P.M. The second command, which is issued after the batch job completes, rerecords the data by creating a new version of the MONITOR.DAT file, containing only the DECnet-VAX data.

12.6 USAGE CONSIDERATIONS

This section contains information to help you run the utility effectively.

12.6.1 File Space

When recording is active (or display output is being routed to a disk file), it is possible to consume large quantities of disk space in a short period of time. In particular, if disk quota is exceeded during execution of a MONITOR request, open files are closed and the request is terminated prematurely. To avoid this situation, carefully plan recording requests by estimating the amount of disk space required, using the following rule of thumb: allow 70 bytes per class per interval or, for the PROCESSES class, 70 bytes per process per interval. Then check the amount of disk quota available, and set appropriate values for /INTERVAL and /ENDING.

If necessary, refer to Appendix D, MONITOR Recording File Record Formats, for details on the exact record sizes for this version of VAX/VMS MONITOR.

12.6.2 Time to Perform Collection can be Significant

The collection interval, specified in a live collection MONITOR request with the /INTERVAL qualifier, is defined as the number of seconds from the end of one collection event to the beginning of the next. A collection event includes collection for all requested classes. So, the elapsed time from the beginning of one collection event to the beginning of the next is the interval value plus the time it takes to do the collection. For some requests, notably those including many classes or the PROCESSES class, collection time can be significant.

12.6.3 Specifying /INTERVAL for Playback

To understand how to use the /INTERVAL qualifier for playback, you must be aware of various events occurring periodically within a MONITOR request. These are the collection event, the recording event and the display event.

The /INTERVAL qualifier is the primary means of controlling the frequency of these events. When a collection event occurs, raw data for all requested classes is collected from the operating system or from a previously recorded file. When a recording event occurs, data for all requested classes is written to a recording file. When a display event occurs, a screen image is composed, for a single class, from the accumulated data collected for that class since the beginning of the MONITOR request.

For live collection requests, a collection event is always followed immediately by a recording event (if requested). The frequency of collection-recording event pairs is controlled by the /INTERVAL qualifier, which specifies the number of seconds that must elapse between occurrences of the event pair. Display events occur asynchronously to collection-recording event pairs, at a frequency governed by the /VIEWING_TIME qualifier.

For playback requests, a collection event occurs each time a new interval is encountered in the input file of previously recorded data. A recording event (if requested) does not necessarily follow immediately as in the case of live collection. Its frequency is still governed by the /INTERVAL qualifier; the specified /INTERVAL value is interpreted in terms of the /INTERVAL value specified when the input file was created. The new value must be an integral multiple of the original value. A recording event is then triggered every time an interval is encountered in the input file that is the appropriate multiple of the original interval.

For playback requests, occurrences of display events (if requested) are indicated in exactly the same way as recording events -- with the /INTERVAL qualifier -- and immediately follow recording events (if both are specified). The actual length of time a displayed image remains on the screen is still specified with the /VIEWING_TIME qualifier, but, unlike the live collection case, this qualifier is not used to signal a display event. Table 12-1 summarizes which qualifiers cause the various MONITOR events.

Table 12-1: Relationship of MONITOR Command Qualifier to Event

Event	Live Collection	Playback
Collection	/INTERVAL	Original /INTERVAL value (from file)
Recording	/INTERVAL	/INTERVAL
Display	/VIEWING_TIME	/INTERVAL

12.7 ERROR MESSAGES

The VAX/VMS System Messages and Recovery Procedures Manual lists the messages generated by MONITOR and provides explanations and suggested user actions.

CHAPTER 13

PERSONAL MAIL UTILITY (MAIL)

The Personal Mail Utility (MAIL) allows you to send messages to other users on your system or on any other VAX-11 computer that is connected to your system by means of DECnet-VAX. You can also file, forward, delete, print, and reply to messages that other users send to you.

Messages that you receive are stored in files called message files. Your default message file, called MAIL.MAI, is created in your default directory the first time you receive mail. New messages that you receive are appended to the end of MAIL.MAI. You can copy messages to other message files (which all have the default file type MAI) that you name using the FILE command.

Figure 13-1 shows a sample 3-message file.

```
                                                            MAIL #1
From:  HDV                     23-JUN-1982  16:25
To:    @GROUP
Subj:  Change of Location

Next Tuesday's review meeting has been moved from Conference Room
A to Conference Room D (on the second floor).  See you there.
                                                            MAIL #2
From: MENDOZA                  24-JUN-1982  10:06
To:   MALCOLM
Subj: Payroll program

There is a new version of the payroll program in my directory:
[MENDOZA]PAYROLL.FOR;12.
                                                            MAIL #3
From: NOEL                     28-JUN-1982  11:46
To:   MALCOLM
Subj: Source statement format

You may include as many spaces or tabs as you want!
```

Figure 13-1: MAIL Message File

13.1 INVOKING MAIL

The DIGITAL Command Language (DCL) command MAIL can be used either to invoke the MAIL Utility or if specified with parameters, to send a file to another user and return you to DCL (see Section 13.3).

When MAIL sends you a message from another user and you are logged in, MAIL notifies you with a message on your terminal. For example:

```
     New Mail from CARLSON
```

You will also be notified that you have new mail when you log in and when you invoke the MAIL Utility.

To invoke MAIL, enter the following command in response to the DCL prompt

```
     $ MAIL
```

The utility responds with the prompt:

```
     MAIL>
```

You can now issue the commands described in Section 13.2 to send and display messages.

Figure 13-2 shows typical uses of the MAIL Utility, including reading, sending, deleting, and filing messages, and creating new message files.

13.2 MAIL COMMANDS

MAIL commands consist of one word typed in response to the prompt MAIL>. These commands can be abbreviated to a unique, shorter form (usually as short as one letter). Note that D is the short form of DELETE (not DIRECTORY) and R is the short form of REPLY (not READ).

Table 13-1 summarizes the MAIL commands, which are described in the following sections.

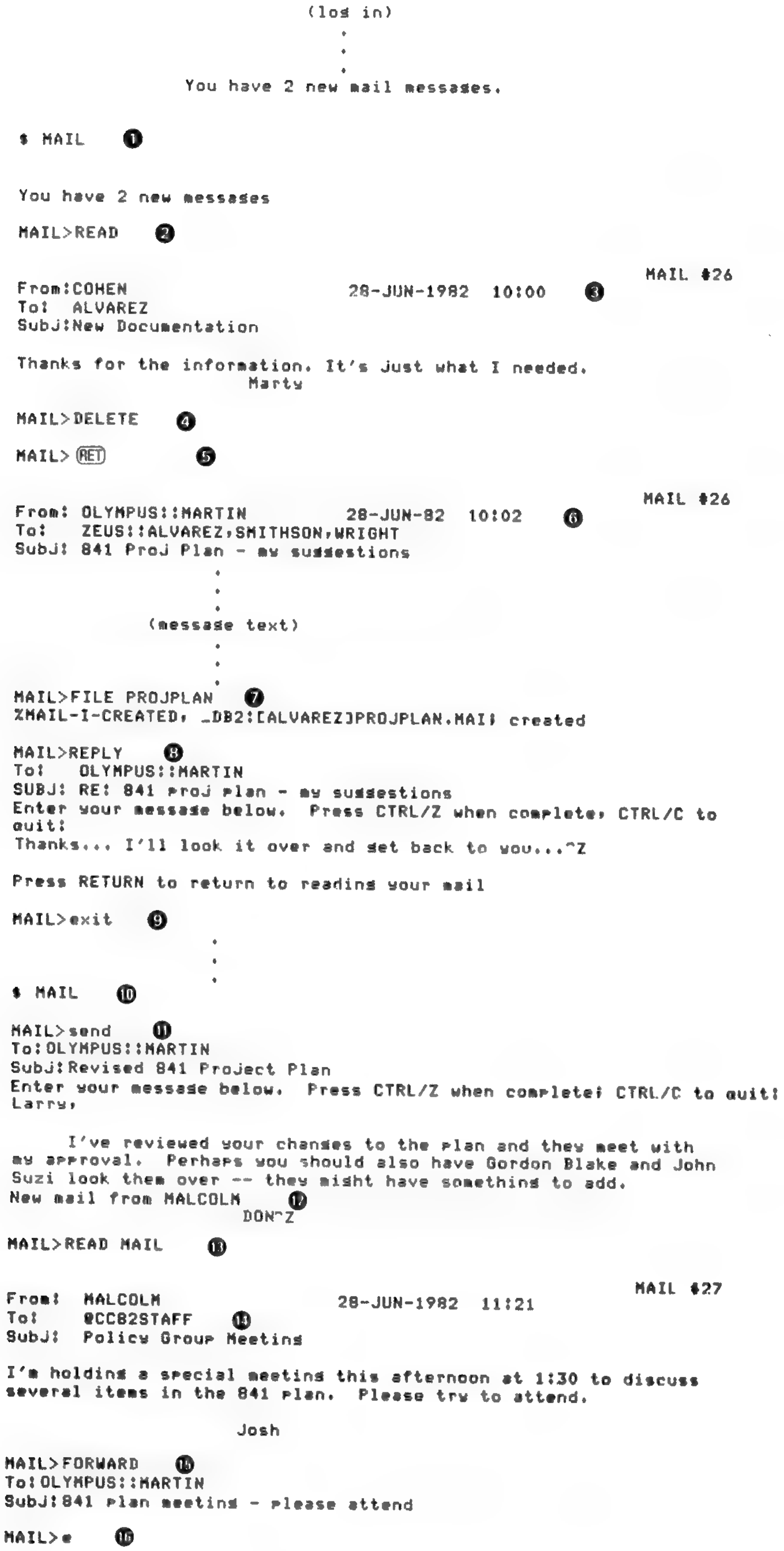
Table 13-1: Summary of MAIL Commands

Command	Meaning
BACK	Backs up to the previous message
DELETE	Deletes the current (last-read) message
DIRECTORY	Lists a summary of your messages

(continued on next page)

Table 13-1 (Cont.): Summary of MAIL Commands

Command	Meaning
EXIT	Exits from MAIL (you can also use CTRL/Z to exit from MAIL)
FILE	Copies current (last-read) message into a specified file
FORWARD	Forwards current (last-read) message to user or users
HELP	Displays information on how to use MAIL
NEXT	Skips to the next message
PRINT	Prints the current (last-read) message
QUIT	Cancels message deletions and exits from MAIL leaving the message file exactly as it was when you entered MAIL
READ and RET	Displays next page, next message, or (READ only) specified message
REPLY	Sends a reply to the sender of the current (last-read) messsage
SEARCH	Searches for a message that contains specified text
SEND	Sends a message to a user or users



```
                    (log in)
                       .
                       .
                       .
          You have 2 new mail messages.


$ MAIL  ❶


You have 2 new messages

MAIL>READ  ❷

                                                        MAIL #26
From:COHEN                        28-JUN-1982  10:00  ❸
To:  ALVAREZ
Subj:New Documentation

Thanks for the information. It's just what I needed.
                     Marty

MAIL>DELETE  ❹

MAIL> (RET)       ❺

                                                        MAIL #26
From: OLYMPUS::MARTIN             28-JUN-82  10:02  ❻
To:   ZEUS::ALVAREZ,SMITHSON,WRIGHT
Subj: 841 Proj Plan - my suggestions
                    .
                    .
                    .
            (message text)
                    .
                    .
                    .
MAIL>FILE PROJPLAN  ❼
%MAIL-I-CREATED, _DB2:[ALVAREZ]PROJPLAN.MAI; created

MAIL>REPLY    ❽
To:   OLYMPUS::MARTIN
SUBJ: RE: 841 proj plan - my suggestions
Enter your message below.  Press CTRL/Z when complete, CTRL/C to
quit:
Thanks... I'll look it over and get back to you...^Z

Press RETURN to return to reading your mail

MAIL>exit    ❾
                    .
                    .
                    .
$ MAIL   ❿

MAIL>send    ⓫
To:OLYMPUS::MARTIN
Subj:Revised 841 Project Plan
Enter your message below.  Press CTRL/Z when complete; CTRL/C to quit:
Larry,

     I've reviewed your changes to the plan and they meet with
my approval.  Perhaps you should also have Gordon Blake and John
Suzi look them over -- they might have something to add.
New mail from MALCOLM     ⓬
                     DON^Z

MAIL>READ MAIL    ⓭

                                                        MAIL #27
From:  MALCOLM                    28-JUN-1982  11:21
To:    @CC82STAFF    ⓮
Subj:  Policy Group Meeting

I'm holding a special meeting this afternoon at 1:30 to discuss
several items in the 841 plan.  Please try to attend.

                     Josh

MAIL>FORWARD   ⓯
To:OLYMPUS::MARTIN
Subj:841 plan meeting - please attend

MAIL>e   ⓰
```

Figure 13-2: Sample MAIL Session

❶ Invokes MAIL Utility.

❷ Reads first new message in default message file, MAIL.MAI. Pressing RETURN would have produced the same result.

❸ First new message -- number 26 in the MAIL.MAI file.

❹ Deletes message #26 from the MAIL.MAI file. You must be reading a message in order to delete it.

❺ RETURN advances to the next new message received. Note that, since previous message #26 was deleted, this message becomes #26.

❻ Message #26 is from MARTIN on node OLYMPUS to the reader ALVAREZ on node ZEUS and to SMITHSON and WRIGHT on node OLYMPUS.

❼ ALVAREZ decides to create a new file, PROJPLAN, in which to store project plan messages.

❽ ALVAREZ replies to MARTIN concerning message #26. You must be reading a message in order to reply to it.

❾ Exits from MAIL Utility. If RETURN had been pressed instead, message #26 would have been redisplayed.

❿ Invokes MAIL Utility.

⓫ Initiates SEND function. MAIL prompts with the To:, Subj:, and Enter your message... lines which the sender completes.

⓬ Notice of new mail while in the process of sending mail to user MARTIN on node OLYMPUS. Sender chooses to complete the message.

⓭ Reads new mail, which becomes message #27 in MAIL.MAI file.

⓮ Message #27 is to the users, including ALVAREZ, on the CC82STAFF distribution list (see Section 13.2.13.3).

⓯ Forwards message #27 to MARTIN. MAIL prompts with the To: and Subj: lines. The sender completes these lines and the current message is forwarded. You must be reading a message in order to forward it.

⓰ Exits from MAIL Utility.

13.2.1 BACK Command

The BACK command displays the message preceding the current (last-read) message.

13.2.2 DELETE Command

The DELETE command deletes the current (last-read) message.

You must be reading a message in order to delete it. For example:

```
MAIL> READ

                                                          MAIL#4
From:       CARLSON                29-JUN-82 19:15
To:         MALCOLM
Subj:       Monthly Meetings
                  .
                  .
                  .
    (message text)
                  .
                  .
                  .

MAIL> DELETE
```

Although this message is apparently deleted when you type DELETE, it is not actually deleted from the file until you exit from MAIL or read messages from another file. Thus, if you accidentally issue the DELETE command to delete a message, you can override the DELETE command with the QUIT command (see Section 13.2.10).

13.2.3 DIRECTORY Command

The DIRECTORY command lists a summary of the messages in the current or specified message file, including message number, sender's name, date, and subject:

```
DIRECTORY [filename]
```

filename

A message file. If a file name is specified, MAIL displays a summary of the messages in the file. If no file name is specified, MAIL displays a summary of the messages in the current file, as follows. The current message file contains all messages that are currently present in the file. Deleted messages are no longer contained in the current message file.

```
MAIL> DIRECTORY

                                              MAIL

    # From           Date          Subject

    1 MURRAY          7-JUN-1982   FIELD TEST
    2 ZEUS::PETER    24-JUN-1982   Code Review
    3 MARCEL         26-JUN-1982   Monthly Meetings
    4 LDAVIS         26-JUN-1982   PASCAL
    5 JOSH           26-JUN-1982   LINKER
```

13.2.4 EXIT Command

The EXIT command allows you to exit from the MAIL utility.

```
MAIL> EXIT

$
```

You can also exit from MAIL by typing CTRL/Z.

13.2.5 FILE Command

The FILE command is used to save a copy of the current (last-read) message in a specified message file. The copy is appended to the end of the specified file.

```
     FILE filename
```

filename

> The message file in which the current message is to be saved. If the specified message file does not exist or you do not give a complete file specification, MAIL creates a file in your default directory, giving it the file type MAI.

You must be reading a message in order to file it.

13.2.6 FORWARD Command

The FORWARD command sends a copy of the current (last-read) message to a user or users. MAIL prompts you for the name of the user or users to whom you want to forward the message. See the SEND command for more information on sending messages.

You must be reading a message in order to forward it.

13.2.7 HELP Command

The HELP command allows you to obtain information about the MAIL utility.

To obtain information about all of the MAIL commands, type

```
     HELP *
```

To obtain information about individual commands or topics, type HELP followed by the command or topic name. For example:

```
     MAIL> HELP BACK

     BACK

       Displays the previous message (the message before the current
       message).

     MAIL>
```

13.2.8 NEXT Command

The NEXT command skips to the next message and displays it. This command is useful if, while reading through your messages, you encounter a particularly long message that you would like to skip over.

13.2.9 PRINT Command

The PRINT command queues a copy of the current (last-read) message for printing. The file(s) created by the PRINT command are not actually released to the print queue until you exit from MAIL, so that multiple messages will be concatenated into one print job. The PRINT command takes an optional qualifier, as follows:

```
     PRINT [/QUEUE=queue-name]
```

/QUEUE=queue-name

> The device on which a message is to be printed. If the qualifier is not specified, the last queue name specified during the current session is used. If an explicit queue name has never been specified, SYS$PRINT is used.

You must be reading a message in order to print it.

13.2.10 QUIT Command

The QUIT command cancels message deletions and exits from MAIL. Messages selected for deletion by the DELETE command are not deleted from the MAIL file.

```
     QUIT
```

Thus, if you accidentally issue the DELETE command to delete a message, you can cancel the DELETE command by issuing the QUIT command before exiting from MAIL or reading messages from another file. (QUIT performs the same function as CTRL/Y.)

13.2.11 READ Command

The READ command displays your messages. It can be issued with or without parameters, in the following formats:

```
     READ [filename] [message-number]

     message-number

     RET
```

filename

> A message file. File type MAI and your default directory are assumed.
>
> If a file name is specified, MAIL will display messages from that file. If no file name is specified, MAIL will display messages from the current file. The default file when you enter the MAIL utility is MAIL.MAI.

message-number

> A number representing the position of a message in a message file. If you specify a number greater than the number of messages in the file, MAIL will display the last message in the file. Therefore, to read the latest message in a file, specify a very large message number.

You can display a message by entering only its message number, without the READ command.

RET

The RETURN key. Pressing this key is the same as entering the READ command without parameters.

If you issue the READ command without parameters or press RETURN immediately after the MAIL Utility is invoked, MAIL displays the first page of your oldest unread message from your MAIL.MAI file. If there are no unread messages, MAIL displays the oldest message in the file. Each time you enter the READ command without parameters, or press RETURN, MAIL displays the next page, or the next message if there are no more pages in the current message.

If a new message arrives while you are in MAIL, you can type READ MAIL to read the message, and then return to the previous MAIL activity.

The date and time that appear on MAIL messages reflects the receiver's time, not the sender's time.

13.2.12 REPLY Command

The REPLY command sends a message to the sender of the current (last-read) message.

```
REPLY[/qualifier] [file-spec]
```

qualifier

One of the two qualifiers (/EDIT and /LAST) listed in Table 13-2. If you specify both qualifiers, the /EDIT qualifier is ignored. (See Section 13.2.13.1 for information on changing the default editor invoked by the /EDIT qualifier.)

file-spec

A file to be sent as your reply. If no file is specified, you will be prompted for the text of your reply.

You must be reading a message in order to reply to it.

13.2.13 SEND Command

The SEND command sends a message to another user or users. (If you simply want to send a file to another user or group of users, you may want to use the extended form of the MAIL command described in Section 13.3 instead of using SEND.) You can include a file specification in the SEND command.

```
SEND[/qualifier] [file-spec]
```

qualifier

One of the two qualifiers (/EDIT and /LAST) listed in Table 13-2. If you specify both qualifiers, the /EDIT qualifier is ignored. (See Section 13.2.13.1 for information on changing the default editor invoked by the /EDIT qualifier.)

file-spec

a file to be sent. If no file is specified, you will be prompted for the text of your message.

MAIL prompts you first for the name of the user or users who will receive the message:

```
To:
```

You reply with the user name(s) or with the file name of a distribution list file (as described in Sections 13.2.13.2 and 13.2.13.3), in the following format:

```
[[nodename::]username,...] [,] [@listname]
```

Then MAIL prompts you for the subject of the mail:

```
Subj:
```

If you specify a file in the SEND command, the text in that file is sent to the users.

If you do not specify a file, MAIL displays the following prompt:

```
Enter your message below.  Press CTRL/Z when complete, CTRL/C  to
quit.
```

Type the message that you want to send; then press CTRL/Z. Note that once you have typed a line and pressed RETURN, there is no way to change it. If you decide not to send a message you are typing, press CTRL/C to abort the message. You will then receive the MAIL> prompt.

You can substitute a logical name for the name of the user to whom you send a message. You must assign the logical name before entering MAIL, for example:

```
$ DEFINE JOHN  DOE
```

Once in MAIL, you can answer the "To:" prompt to the SEND command with JOHN rather than DOE. Section 13.2.13.2 describes the use of logical names in messages sent by way of DECnet-VAX to a user many nodes removed from the sender's node.

Table 13-2 and Section 13.2.13.1 provide instructions on invoking a text editor to edit your messages.

Table 13-2: SEND and REPLY Qualifiers

Qualifier	Function
/EDIT	Specifies that a text editor is to be called to edit the message that you are sending. If you have included a file specification in the command, that file will be opened for editing. If you have not specified a file, the editor will be invoked so that you can edit your new message. The file created, MAIL.TMP, is deleted when the message is sent.

(continued on next page)

Table 13-2 (Cont.): SEND and REPLY Qualifiers

Qualifier	Function
	An output file can be specified in the form SEND/EDIT/OUTPUT=filename. The command that you use to exit from the editor will complete the SEND or REPLY operation. For this qualifier to work properly, your default command interpreter must be DCL.
/LAST	Specifies that the last message that you sent should be used as the text for the message. The /LAST function remains in effect only during the current MAIL session. The /EDIT qualifier is ignored if /LAST is specified.

13.2.13.1 Changing the Default Editor Invoked by the /EDIT Qualifier - If the logical name MAIL$EDIT is defined, its equivalence name will be used as the name of a command procedure that will invoke the editor. Note that because the command procedure defined by MAIL$EDIT is executed in the context of a subprocess, the definition of MAIL$EDIT and the command procedure itself must not refer to any process logical names defined by the initiating process.

If MAIL$EDIT is not defined, the command procedure SYS$SYSTEM:MAILEDIT.COM will be called. This command procedure contains the DCL command EDIT, which invokes the EDT text editor.

To change the default editor for MAIL (for example, from EDT to SOS) you must copy the MAIL$EDIT command procedure to your directory and then modify it. First, enter the following command line:

```
     $ COPY SYS$SYSTEM:MAILEDIT.COM MAILEDIT.COM
```

This copies the command the system uses to your directory.

Next, edit MAILEDIT.COM by changing "EDIT" to "EDIT/SOS." The resulting command procedure looks like this:

```
     $ !++
     $ ! Default command procedure to invoke an editor for MAIL.
     $ !
     $ ! Inputs:
     $ !
     $ !         P1 = Input file name
     $ !         P2 = Output file name
     $ !
     $ ! Note that this procedure is run in the context of a sub-
     $ ! process, therefore LOGIN.COM is not executed, creator
     $ ! process logical names do not exist, and the default
     $ ! directory is the same as the creator process.
     $ !--
     $ DEFINE/USER SYS$INPUT 'F$LOGICAL("SYS$OUTPUT")'
     $ IF P1 .EQS. "" THEN GOTO NOINPUT
     $ EDIT/SOS/OUTPUT='P2' 'P1'
     $ EXIT
     $ NOINPUT:
     $ EDIT/SOS 'P2'
```

Finally, enter the following line in your LOGIN.COM file:

```
     DEFINE MAIL$EDIT disk:[directory]MAILEDIT.COM
```

where disk: is the disk on which the file is located and [directory] is your directory name.

13.2.13.2 Sending Messages via DECnet-VAX

If you include a node name with the user name, the message is sent by means of DECnet-VAX to that user (if the node name is omitted, MAIL assumes that the user is on your node):

```
     nodename::username
```

nodename::

The node at which the user who is to receive the message is located.

username

The name of the user who is to receive the message.

You can specify node names and user names as logical names. For example, if the user Arthur King is on node KAMLOT and you did not assign a logical name, you would have to type:

```
     To: KAMLOT::KING
```

However, if you had previously made the assignment:

```
     $ DEFINE ART KAMLOT::KING
```

you could simply respond to the prompt with the logical name ART:

```
     To: ART
```

The DEFINE command could be included in your LOGIN.COM file, along with other DEFINE commands for users to whom you frequently send messages by way of DECnet-VAX.

13.2.13.3 Sending Messages to Distribution Lists

If you frequently send mail to the same group of users, you may find it helpful to use a distribution list. A distribution list is a file containing the names of users to whom you want to send messages.

To set up a distribution list, use the DCL command EDIT or CREATE to create a distribution list file with the file type DIS. Enter one user name per line in this file. A distribution list can also include the names of other distribution lists; the depth to which you can nest distribution lists is determined by your Open File Quota. You can include comments by entering lines whose first character is an exclamation point (!). For example:

```
     $ CREATE WRITERS.DIS
     !SOFTWARE WRITERS:
     PIERSON
     NODE3::JOSEPHS
     LAWRENCE
     NODE4::ASHLEY
     !NESTED DISTRIBUTION LIST:
     @STAFF
```

To use the distribution list file, you enter its file name preceded by an at sign (@) in response to the To: prompt. For example:

```
     To:  @WRITERS
```

You can enter separate user names along with the distribution list if the distribution list is the last entry. For example:

```
     To:  GEORGE, MARCEL, BEN, ERICA, @WRITERS
```

However, you cannot send mail to more than one distribution list at one time (unless they are nested). For example, MAIL will not accept the following command:

```
     To:  @WRITERS, @OPERATORS
```

13.2.14 SEARCH Command

The SEARCH command searches for messages that contain a specified text string.

```
     SEARCH [search-string]
```

search-string

> The text string that is searched for in your current message file. The search starts from the beginning of the file. If search-string is not specified, a search is made for the previously specified string, starting after the current (last-read) message.

13.3 USING THE DCL COMMAND MAIL TO SEND FILES

By specifying parameters when you enter the DCL command MAIL, you can use a single command line to send a file to one or more users. When the MAIL command is used with parameters, the command string has the following format:

```
     MAIL[/SUBJECT="text"] file-spec [username[,...]] ["@listname"]
```

text

> The subject of the message. If you include more than one word, you must enclose the text in quotation marks.
>
> If you omit the /SUBJECT qualifier, the message is sent without a subject notation.

file-spec

> A file containing message text to be sent. If you omit the file type, the default file type is TXT. No wild card characters are allowed in the file specification.
>
> If you do not specify a file in the command string, the MAIL Utility is invoked, as shown above.

username[,...]

> One or more users to receive the message. If you do not specify a user name in the command string, you will be prompted for the user name.

"@listname"

The name of a distribution list file. The default file type is DIS. Setting up a distribution list file is described in Section 13.2.13.3. The quotation marks and at sign (@) are required. A distribution list name that follows a user name specification must be preceded by a comma.

Examples:

```
$ MAIL/SUBJECT="for your information" MEETING THOMAS,SLOAN
```

This command string contains the subject text ("for your information"), file-spec (MEETING), and username (THOMAS,SLOAN) parameters. Because the file type was omitted, MAIL searches your default directory for the file MEETING.TXT.

```
$ MAIL NOTICE "@WRITERS"
```

This command string contains the file-spec (NOTICE) and listname ("@WRITERS") parameters. Because the /SUBJECT qualifier was not included, the message is sent without a subject notation. MAIL assumes the default file type TXT for the file NOTICE.

13.4 PROTECTION OF MAIL FILES

MAIL files are protected by a code, or mask, that is the same as the code used in protection other types of files. Two types of user (defined by UIC) can have access to mail files: system and group.

13.5 SYSTEM MANAGEMENT AND MAIL

In order for the MAIL Utility to deliver a message to a SYSTEM, FIELD, or SYSTEST user, the system manager must first add a default device name to the authorization file of each of these users.

Mail messages are stored in the user's default login directory as ASCII text, one line per record, in standard variable length record files. Each message is delimited by two records, the first containing only the ASCII form-feed character, and the second beginning with the word "From:". Each line of text can contain a maximum of 255 characters.

A count of the number of new messages that a user has received is stored in the user's system authorization record in SYS$SYSTEM:SYSUAF.DAT. This count is used to make up the message that you receive upon logging in if you have new mail messages waiting to be read. If the user authorization file is replaced with a copy (for example, a back-up copy), the count in the file, and therefore the message received upon logging in, may not correspond to the actual mail messages in user's mail files. This inconsistency disappears, however, the first time a message file is read in its entirety.

Mail keeps SYSUAF.DAT open while MAIL is being run, sometimes preventing this file from being copied. (The Convert Utility can be used to copy SYSUAF.DAT.)

In MAIL messages sent via DECnet-VAX, the user can specify node names and user names as logical names. They are translated like VAX-11 RMS specifications: a node name or user name is only translated if it is the first string in the specification. Any access control information in the node name or logical name is ignored.

NOTE

> See the DECnet-VAX System Manager's Guide for information on defining the MAIL network object type.

13.6 MAIL STATUS MESSAGES

The VAX/VMS System Messages and Recovery Procedures Manual lists the messages issued by MAIL and provides explanations and suggested user actions for these messages.

MAIL messages are in the form:

```
%MAIL-l-message, message-text
```

The l is a severity code, either E for error, W for warning, or I for information. The message is a mnemonic representing the specific error that occurred. The message-text is a brief description of the condition that caused the message to be issued.

For more information on messages, refer to Chapter 11, Message Utility, and to the VAX/VMS System Messages and Recovery Procedures Manual.

CHAPTER 14

PHONE UTILITY (PHONE)

The VAX/VMS Phone Utility (PHONE) allows you to talk with other users on your system or any other VAX-11 computer that is connected to your system by means of DECnet-VAX. It was designed to closely simulate the features of a real telephone, including the hold button, conference calls, and telephone directories.

PHONE can only be used on video terminals with direct cursor positioning, such as the DIGITAL VT52 and VT100.

14.1 INVOKING PHONE

The DIGITAL Command Language (DCL) command PHONE can be used to invoke the PHONE Utility. To invoke PHONE, enter the following command in response to the DCL prompt:

```
     PHONE[/qualifier(s)] [phone-command]
```

/qualifier(s)
     The qualifiers that modify the characteristics of the simulated telephone.

phone-command
     Any valid PHONE command that is to be executed before PHONE prompts you for additional commands. See Section 14.2 for a description of PHONE commands.

14.1.1 Command Qualifiers

When specifying the DCL command PHONE, you can supply qualifiers that modify the characteristics of the simulated telephone. Table 14-1 describes the valid qualifiers.

Table 14-1: Summary of PHONE Qualifiers

Qualifier	Function
/SCROLL /NOSCROLL	Specifies what your terminal does when the viewport, the area on the screen where a user's conversation is displayed, becomes full. If /SCROLL is specified, the text is scrolled up one line and new text goes on the bottom line. If /NOSCROLL is specified, the new text is wrapped around and appears on the top line of the viewport. Specifying /NOSCROLL will improve the response time on slow-speed terminals. The default is /SCROLL.
/SWITCH_HOOK="c"	Specifies the telephone switch hook character. The switch hook character must be typed before each command to the PHONE Utility. Text typed without this character is considered part of the conversation. The default switch hook character is the percent sign (%).
/VIEWPORT_SIZE=n	Specifies the maximum number of lines in a viewport, including the heading line and the bottom line of dashes. The valid range is 3 to 10, with a default of 10. Smaller viewports will improve the response time on slow-speed terminals.

Table 14-2: Summary of PHONE Commands

Command	Meaning
ANSWER	Answers the phone when you receive a call
DIAL	Places a call to another user
DIRECTORY	Displays a list of the users you may call
EXIT	Exits from PHONE
FACSIMILE	Includes the contents of a file in your conversation
HANGUP	Hangs up your own phone
HELP	Displays information on how to use PHONE
HOLD	Places other users on hold
MAIL	Sends a short MAIL message to another user
REJECT	Rejects a call from another user
UNHOLD	Reverses the previous HOLD command

14.2 PHONE COMMANDS

You enter PHONE commands by typing the switch hook character and then a PHONE command. The phone command may be abbreviated. Some commands also require additional information typed after the keyword.

If you are using PHONE but are not currently engaged in a conversation, the switch hook character is optional because there is no ambiguity between a command and conversation.

You can type CTRL/W at any time to restore the screen while continuing your current conversation.

Table 14-2 summarizes the PHONE commands, which are described in the following sections.

14.2.1 ANSWER Command

The ANSWER command is used to answer the phone when you receive a call. If the call comes when you are not currently using the PHONE Utility, a message will be broadcast to your terminal. If you are using PHONE, the message will appear in PHONE's standard message line, which is located under the command input line on your screen (see Figure 14-1).

You can do one of three things when your phone rings:

- Ignore it
- Answer the phone, establishing a link with the caller
- Reject the call (see Section 14.2.10)

14.2.2 DIAL Command

The DIAL command is used to place a call to another user. You must include the user name of the person to whom you wish to place the call. This user name can be preceded by a DECnet node name if the person is on another node. The full form of this command is:

```
DIAL [node::]user-name
```

For example:

```
$ DIAL TAURUS::SMITH
```

A logical name may be substituted for the parameter.

When you enter this command, PHONE broadcasts a message to Smith's terminal that indicates you are calling. This message will flash every ten seconds until one of the following happens:

- Smith answers the phone.
- Smith rejects the call.

NOTE

> If you enter a user name without any command, the DIAL command is assumed.

14.2.3 DIRECTORY Command

The directory command lists those users with whom you can talk on your system or any other system in a network. If you enter the command without additional information, it will list the users on your system. If you enter the command with a node name, it lists the users on that system. You may substitute a logical name for the node name in the command line. The complete form of the DIRECTORY command is:

```
     DIRECTORY [node [::] ]
```

The directory is displayed on your terminal, line by line, until the entire list is displayed or until you type any key on the keyboard. The following information is displayed about each user:

- The process name and user name
- Whether or not the user's terminal can be used as a telephone
- The status of the user's simulated telephone

14.2.4 EXIT Command

The EXIT command is the standard VAX/VMS command for leaving a utility. When you enter this command, PHONE automatically executes the HANGUP command (see Section 14.2.6) and then returns to DCL.

Typing CTRL/Z on the command line, when there is no conversation, is equivalent to entering the EXIT command.

14.2.5 FACSIMILE Command

The FACSIMILE command allows you to include the contents of a file in your conversation. It sends the contents of that file to all current participants in a call. The complete syntax is:

```
     FACSIMILE file-spec
```

PHONE continues to send the file until it reaches end of file or until you type any key on your keyboard.

14.2.6 HANGUP Command

The HANGUP command is used to hang up your own phone. This disconnects all current links -- the current conversation, everyone you have on hold, and anyone who has you on hold.

Typing CTRL/Z during a conversation is equivalent to entering the HANGUP command.

14.2.7 HELP Command

The HELP command allows you to obtain information about the PHONE Utility. To obtain general information about the utility, type:

```
     HELP
```

To obtain specific information about individual commands or topics, type HELP followed by the command or topic name:

```
     HELP topic
```

The information you request is displayed at your terminal until all information on the selected topic is displayed or until you type any character at your keyboard.

14.2.8 HOLD Command

This command allows you to put other users on hold. When you enter the command, everyone who is currently on the phone with you (including anyone who has you on hold) is placed on hold. Each person is informed that he has been placed on hold.

PHONE allows you to make calls to other users while you have someone on hold.

14.2.9 MAIL Command

The MAIL command allows you to send a message to another person. For example, if you tried to call someone who was not in, you could leave a message to call you back.

The MAIL command requires two parameters. The first is the user name of the person to whom the message is to be sent. The second is the message, enclosed in quotation marks ("). The complete syntax of the command is:

```
     MAIL [node::]user-name "short message"
```

A logical name may be substituted for the first parameter.

14.2.10 REJECT Command

The REJECT command is used to reject a phone call from another user. The user will receive a message at his terminal that the call has not been accepted.

The REJECT command will accept the optional parameter EXIT. If this parameter is specified, PHONE performs an EXIT command following the rejection. This allows you to set up a personalized VAX/VMS command that can reject a call and return to DCL without your intervention. Such a personalized command might appear in your login command file as:

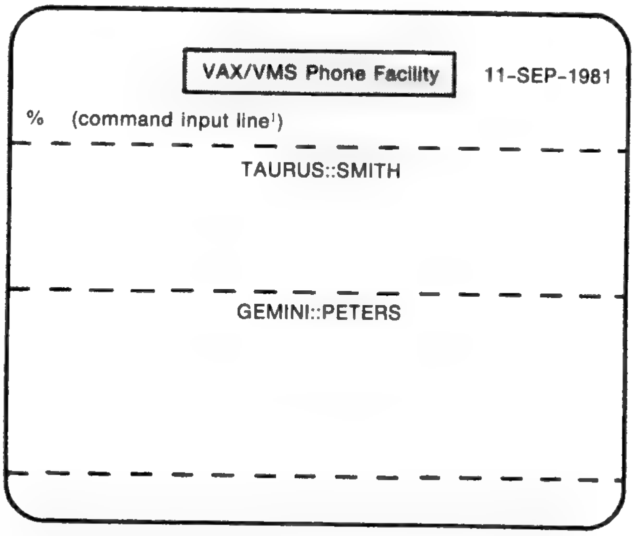
```
     REJECT :== "PHONE REJECT EXIT"
```

14.2.11 UNHOLD Command

The UNHOLD command reverses the previous HOLD command. A user who was placed on hold by this previous command is removed from hold, replacing any prior conversation. A user who was placed on hold before the previous HOLD command remains on hold.

14.3 SCREEN LAYOUT

PHONE can only be used on video terminals with direct cursor positioning, such as the DIGITAL VT52 and VT100. When PHONE is invoked it takes over the entire screen and formats it as shown in Figure 14-1.



1. The switch hook character is always displayed in column 1 of the command input line.

ZK-898-82

Figure 14-1 PHONE Format

Each person engaged in the conversation has a viewport on the screen. The viewport is the area in which information regarding each user is placed. This information consists of the person's name and the text of their conversation, along with various status indicators. The status indicators include such information as who is on hold.

PHONE displays at most six viewports at any time. People you have on hold may be temporarily eliminated from the screen to make room for new participants.

14.4 CONVERSATION TEXT

While talking to one or more people, most of the characters that you type are considered part of the conversation and are sent to each participant. The exception is the switch hook character, which signals that you want to enter a command (you may enter any of the commands in Table 14-2 while engaged in conversation).

All normal ASCII text characters are simply displayed in the viewport. Some of the control characters can be used for formatting, as shown in Table 14-3. Any other control character is ignored.

Table 14-3: Summary of Formatting with Control Characters

Character	Formatting Function
delete	Deletes previous character
line feed	Deletes previous word
return	Starts a new line in the viewpoint
tab	Advances to next tab stop
CTRL/G	Sounds the buzzer at both your terminal and the terminal of anyone with whom you are having a PHONE conversation
CTRL/L	Clears all text in the viewport
CTRL/Q	Allows normal receiving of characters; negates a CTRL/S
CTRL/S	Freezes the characters currently on screen
CTRL/U	Clears current viewpoint line
CTRL/W	Restores entire screen
CTRL/Z	Equivalent to a HANGUP command if typed during conversation; equivalent to an EXIT command if typed when no conversation is in progress

14.5 CONFERENCE CALLS

When setting up a conference call, one participant should act as the operator. This person should set up the conference by calling the other participants in turn and waiting for an answer.

Only users currently participating in the conference call can bring in new participants with the DIAL command. If you dial a person already participating in a conference call, you will talk only to the participant you called.

14.6 USE OF LOGICAL NAMES

The DIAL, DIRECTORY, and MAIL commands accept logical names. If you want to prevent PHONE from treating the parameter to these commands as logical names, prefix the parameter with an underscore (_).

CHAPTER 15

RMS SHARE UTILITY (RMSSHARE)

The RMS Share Utility (RMSSHARE)is used primarily as a system management tool to perform the following functions:

- Enable the VAX-11 Record Management Services (VAX-11 RMS) file sharing capability by initializing file sharing structures in paged dynamic memory (system S0 space), and set the maximum number of pages that the structures can occupy. The VAX-11 RMS file sharing capability must be enabled each time the operating system is booted.

- Display figures on allowable and actual usage (provided VAX-11 RMS file sharing has already been enabled), and increase the maximum number of pages that the file sharing structures can occupy.

15.1 INVOKING, RUNNING, AND TERMINATING RMSSHARE

You need the CMEXEC privilege to run RMSSHARE. To invoke the utility, enter the following command:

```
$ RUN SYS$SYSTEM:RMSSHARE
```

If VAX-11 RMS file sharing is not enabled, RMSSHARE displays the following message (where n represents pages):

```
RMS file sharing is not currently enabled...
  Maximum allocation allowed: n
```

If VAX-11 RMS file sharing is already in effect, RMSSHARE displays the following message sequence:

```
RMS file sharing is currently enabled...
  Maximum allocation allowed: n
  Number of pages allocated: n
  Max pages used: n
  Current number of pages in use: n
```

Here the maximum allocation allowed represents the amount of paged dynamic memory divided by two. The number of pages allocated is the last value set for max pages. Max pages used represents the maximum number of pages used by the system since it was last booted. Regardless of whether or not file sharing is enabled, RMSSHARE then prompts for a maximum page count as follows:

```
Enter max pages:
```

You can do either of the following:

- Enter the maximum number of pages that file sharing structures can occupy. Specify a positive integer not less than the current number of pages allocated and not greater than the maximum number of pages allowed

- Enter the word EXIT (or simply press the RETURN key) to terminate the utility

RMSSHARE continues to display and prompt as shown above until you terminate it.

The system imposes a maximum of one-half the total space in system paged dynamic memory (defined by the PAGEDYN system parameter) for use by VAX-11 RMS file sharing structures.

If you enter an invalid value for the maximum number of pages that file structures can occupy, the following message appears:

```
Invalid size parameter, set to maximum value:  n
```

That is, RMSSHARE allocates the maximum allowable number of pages instead of the invalid value.

15.2 GUIDELINES FOR ESTIMATING SIZE OF FILE SHARING STRUCTURES

You should use the following guidelines to estimate the maximum number of pages required for file sharing structures:

- System base requirement -- 2 pages

- For each sequential file being shared -- 1 page for the first three sharers and 1 page for each additional four sharers

- For each relative file being shared -- 1 page for the first three sharers and 1 page for each additional four sharers

- For each indexed file being shared -- 1 page for the first two sharers and 1 page for each additional two sharers

Total the above pages to obtain an estimate of the space requirements for file sharing structures.

15.3 EXAMPLE

The following example shows how the size of the file sharing structures could be modified from 42 to 44

```
$ RUN SYS$SYSTEM:RMSSHARE
RMS file sharing is currently enabled...
  Maximum allocation allowed: 64
  Number of pages allocated: 42
  Max pages used: 16
  Current number of pages in use: 12
Enter max pages: 44
RMS file sharing is currently enabled...
  Maximum allocation allowed: 64
  Number of pages allocated: 44
  Max pages used:16
  Current number of pages in use: 12
Enter max pages: EXIT
```

15.4 ADDITIONAL SYSTEM ADJUSTMENTS REQUIRED

The site-independent start-up command procedure described in the VAX/VMS System Management and Operations Guide runs RMSSHARE with a specification of 20 for the maximum number of pages for file sharing structures. This number should be adjusted according to the above guidelines.

Additionally, the SYSMWCNT (system maximum working set system parameter) should be set high enough to minimize paging of the VAX-11 RMS file sharing structures in paged dynamic memory. You can use the MONITOR PAGE command of the Monitor Utility to observe the system fault rate and make adjustments as necessary.

You should also adjust the size of the paged dynamic pool (PAGEDYN) system parameter, which is also described in the VAX/VMS System Management and Operations Guide. You use the SYSGEN Utility, described in Chapter 18, to adjust the PAGEDYN system parameter.

15.5 ERROR MESSAGES

The VAX/VMS System Messages and Recovery Procedures Manual lists the messages issued by the RMSSHARE Utility, and provides explanations and suggested user actions.

CHAPTER 16

SLP AND SUMSLP EDITING UTILITIES

Two batch-oriented text editors run on VAX-11 processors: SLP and SUMSLP. To use either of these editing utilities, you generate a list of the changes that you want to apply to your source file. The utilities effect these changes and produce an edited output file. This chapter describes how to use these editors.

16.1 SLP

SLP is a batch-oriented editing program used for source file maintenance. The term "SLP" is an acronym for "source language input program." SLP allows you to update (delete, replace, add) lines in an existing file. Furthermore, SLP gives you a record (audit trail) of editing changes. The SLP command file provides a reliable way to duplicate the changes made to a file, at a later time or on another computer system.

Input to SLP consists of (1) an input source file that you want updated, and (2) a command file containing text lines and edit command lines that specify the update operations to be performed. SLP locates lines to be changed by means of "locators" (sequence numbers or character strings).

SLP output is an updated copy of the input source file. SLP provides an audit trail that helps you keep track of the update status of each line in the file. The audit trail is included permanently in the output file. When a given file is updated with successive versions of an SLP command file, you can use different audit trails to differentiate among the changes made at different times.

SLP output qualifiers modify the appearance of the output file. They let you truncate lines, create or suppress an audit trail, eliminate an existing audit trail, create checksums, and specify the length and beginning position of the audit trail.

16.1.1 Invoking SLP

You can run SLP in either batch mode or interactive mode. In either case, you invoke SLP with the command line:

```
     EDIT/SLP [/qualifiers(s)] infile-spec
```

qualifier(s)

Actions to be performed by SLP that control the generation and format of the listing and output files (see Section 16.1.5).

infile-spec

The input source file specification (see Section 16.1.2.1).

When you run SLP in batch mode from a SLP command file, this command line, preceded by a dollar sign ($), is the first line of the command procedure file, as described in Section 16.1.2.2.

When you run SLP interactively, this command line is typed in response to the DCL prompt ($), as described in Section 16.1.3.

16.1.2 Running SLP In Batch Mode

SLP requires two types of input files: an input source file and an SLP command procedure file. These files are described in Sections 16.1.2.1 and 16.1.2.2.

The output file, described in Section 16.1.2.3, is the permanently updated copy of the input file. It shows the changes SLP makes to the input file.

You can also generate a listing file, as described in Section 16.1.2.3.

Figure 16-1 shows the relationships among the SLP output and input files. The contents of the various files in this figure are described in the following sections.

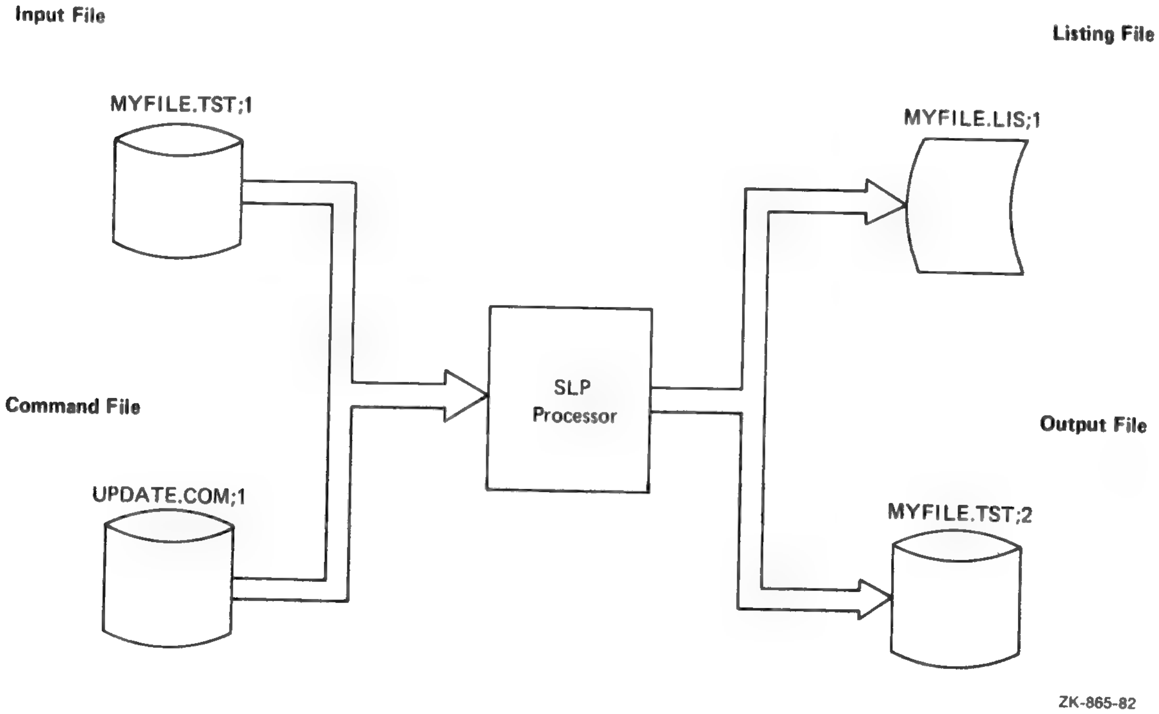


Figure 16-1: Files Used During SLP Processing

16.1.2.1 The Input Source File - The input source file is the file to be updated by SLP. It can contain any number of lines.

To use SLP effectively, you should obtain a sequence-numbered listing of the input file from which you can determine what editing commands you will issue. Section 16.1.5.3 describes how to generate such a listing using the /LIST qualifier. However, the input source file actually updated by SLP can have any kind of line numbers.

16.1.2.2 The SLP Command Procedure File - The SLP command procedure file is a command procedure file that contains SLP editing commands. It consists of four elements:

1. An initialization line that invokes SLP and specifies what file to process:

   ```
   $ EDIT/SLP [/qualifiers] infile-spec
   ```

   This command line is described in Section 16.1.1.

2. SLP editing command lines that define changes to the input file (see Section 16.1.6).

3. Input lines, that is, lines of text that are to be inserted into the output file, either as new lines or to replace old lines.

4. The SLP terminator, a single slash (/) in column 1, that causes SLP to begin its processing (updating) of the file.

An interactive text editor is usually used to create SLP command files. Once you have created the file, you can submit it for processing by using the DCL command Execute Procedure (@) or SUBMIT.

The example below shows a SLP command file named UPDATE.COM. The numbers to the right of the example correspond to the elements listed above.

```
$ EDIT/SLP MYFILE.TST                             (1)
-3                                                (2)
INSERT THIS LINE AFTER LINE 3                     (3)
-4,4                                              (2)
DELETE LINE 4 AND REPLACE IT WITH THIS LINE       (3)
/                                                 (4)
```

You can execute this file by using the Execute Procedure (@) command, as follows:

```
$ @UPDATE
```

Because the file type is the default file type COM, it can be omitted on the DCL command line. (See the VAX/VMS Command Language User's Guide and the VAX/VMS Guide to Using Command Procedures for information on using command procedures and running batch jobs.) When SLP finishes its processing, the DCL prompt ($) is issued.

You can request that SLP calculate a checksum value for SLP editing commands and then use this value to determine whether you have made the correct changes to your source file. See Section 16.1.5.2 for a description of calculating checksums.

16.1.2.3 The Output File - The SLP output file is the updated input file. All of the updates specified by the SLP editing commands are inserted in this file. An audit trail is applied by default, unless suppressed, to new or changed lines (see Section 16.1.4). You can also specify the text and length of an audit trail (see Section 16.1.6.6).

An output file is generated by default. You can suppress the output file, however, by using the /NOOUTPUT qualifier, described in Section 16.1.5.3.

16.1.2.4 The Listing File - You can generate a listing file by using the /LIST qualifier, described in Section 16.1.5.3. The listing file shows the changes made to the source file. Each line in the listing file shows the updates made to the source file. Each line in the listing is numbered in sequence. Updates are indicated by means of an audit trail (unless you suppress audit-trail generation). Section 16.1.4 contains an example of a listing file.

A sequence-numbered listing of the input file can help you determine what editing commands to use. Generating this listing is described in Section 16.1.5.3.

16.1.3 Running SLP Interactively

To use SLP interactively, type the following command line in response to the DCL prompt:

```
EDIT/SLP [/qualifier(s)] infile-spec
```

If you do not enter the input source file specification, you will be prompted for it as follows:

```
File:
```

After specifying the input source file specification, enter SLP editing commands, one per line. Then enter the SLP terminator (/) in the first column of the next line. The utility will respond to the terminator with the prompt:

```
SLP>
```

To end the interactive SLP editing session, type CTRL/Z in response to the SLP> prompt.

16.1.4 How SLP Processes Files

This section contains an example that shows how SLP processes files. The example uses the following source input file, named MYFILE.TST;1.

```
ONE
TWO
THREE
FOUR
FIVE
SIX
SEVEN
EIGHT
NINE
TEN
```

This file is to be updated under the control of the following SLP command procedure file, named UPDATE.COM. The editing commands used in the command file are described in Section 16.1.6.

```
$ EDIT/SLP/AUDIT_TRAIL:50/LIST  MYFILE.TST
-3
INSERT THIS LINE AFTER LINE 3
-4,4
DELETE LINE 4 AND REPLACE IT WITH THIS LINE
/
```

Below is the listing file (MYFILE.LST) that results from issuing the command @UPDATE.

```
MYFILE.TST/AU:50.:10.,MYFILE=MYFILE.TST

      1.  ONE
      2.  TWO
      3.  THREE
      4.  INSERT THIS LINE AFTER LINE THREE                ;**NEW**
      5.  DELETE LINE 4 AND REPLACE IT WITH THIS LINE      ;**NEW**
      6.  FIVE                                             ;**-1
      7.  SIX
      8.  SEVEN
      9.  EIGHT
     10.  NINE
     11.  TEN
```

The audit trail, using the default audit trail texts described in Section 16.1.5.1, shows the new lines (;**NEW**) and indicates where lines have been removed (;**-1). In this case, a new line has been added after line 3, and line 4 has been replaced, causing all subsequent lines to be renumbered. The /AUDIT_TRAIL qualifier in the initialization line indicates that the audit trail is to begin at the next tab stop after column 50.

To process the files, SLP writes each line from the input source file into the output file until it reaches a line to be modified, as requested in the command. When SLP reaches a line to be modified, it makes the indicated modification, notes the change in the audit trail, and then continues writing lines to the output file, in sequence, until it encounters another command or reaches the end of the source file.

The output file, MYFILE.TST;2, is as follows:

```
ONE
TWO
THREE
INSERT THIS LINE AFTER LINE 3                      ;**NEW**
DELETE LINE 4 AND REPLACE IT WITH THIS LINE        ;**NEW**
FIVE                                               ;**-1
SIX
SEVEN
EIGHT
NINE
TEN
```

16.1.5 SLP Qualifiers

SLP qualifiers control the generation and format of the listing file and the output file. You can use them to control the audit trail and output options associated with these files. Table 16-1 describes the SLP qualifiers and their functions. The following sections illustrate the use of SLP qualifiers.

Table 16-1: SLP Qualifiers

Format	Function
/AUDIT_TRAIL [:(POSITION:pos,SIZE:len)] /NOAUDIT_TRAIL	These qualifiers let you suppress audit-trail generation, or specify the beginning position and length of the audit trail. The default is to generate an audit trail 8 characters long, starting in column 80 -- that is, /AUDIT_TRAIL:(POSITION:80,SIZE:8). The maximum allowed value for the size parameter is 16. The audit trail starts at the first tab stop after the position given (or defaulted) for the /AUDIT_TRAIL qualifier. Tab stops are set every 8 columns.
/CHECKSUM [:n] /NOCHECKSUM	The /CHECKSUM qualifier requests a checksum calculation for SLP edit commands. If you do not specify a checksum value, SLP reports the calculated checksum at your terminal. If you do specify a value, SLP will generate a diagnostic message if the specified value does not match the checksum calculation. The default is /NOCHECKSUM.
/LIST [:list-file]	The /LIST qualifier causes SLP to produce a sequentially numbered version of the input file with the same file name. The default file type is LST. You can request a different specification for the listing file by using the /LIST:list-file qualifier.
/OUTPUT [:file-spec]	By default, SLP generates an output file with the same file name and file type as the correction input file. Its version number is higher by 1 than the highest version number existing for the input file name and type. You can request a different specification for the output file by using the /OUTPUT:file-spec qualifier.

(continued on next page)

Table 16-1 (Cont.): SLP Qualifiers

Format	Function
	To suppress the output file, specify /OUTPUT=NL:.
/REPORT /NOREPORT	The qualifier /REPORT causes SLP to report any line truncation by audit trails. If line truncation occurs, SLP prints a diagnostic message. If you specify creation of a listing file, a question mark (?) replaces the period (.) in the line number of the truncation line. The default is /NOREPORT.
/TAB_FILL /NOTAB_FILL	The qualifier /TAB_FILL causes SLP to insert tabs at the end of each text line containing an audit trail. The default is to fill such lines with spaces (that is, /NOTAB_FILL). Using /TAB_FILL saves disk space, because fewer tabs than spaces are required to fill the lines in both the output and the listing file.
/TRUNCATE [:position] /NOTRUNCATE	The /TRUNCATE qualifier lets you truncate input lines to the given column position. If you specify /TRUNCATE but omit position, SLP uses the position given (or defaulted) for the /AUDIT_TRAIL qualifier; position is rounded to the next tab stop before use. Set position at or before the start of the old audit trail that you want to delete. Any trailing spaces or tabs after position are also deleted.

16.1.5.1 Using the /AUDIT_TRAIL Qualifier - You may want to change the position of the audit trail if your output device has fewer than 80 columns or if your source lines are all brief. The following example shows the use of the /AUDIT_TRAIL qualifier to specify the position and length of the audit trail. By default, audit trail texts are ;**NEW** for new lines and ;**-n for deleted lines. (See Section 16.1.6.6 for a description of changing the audit trail text.) The input source file for this example is named MYFILE.TST and is made up of the following lines:

```
ONE
TWO
THREE
FOUR
FIVE
```

The SLP command file is as follows:

```
$EDIT/SLP/AUDIT_TRAIL:(POSITION:30,SIZE:10)/LISTING  MYFILE.TST
-2,.+1,/!CHANGE001/
NEW LINE 2
NEW LINE 3
/
```

The following listing file results from SLP processing.

```
MYFILE.TST/AU:30.:10.,MYFILE=MYFILE.TST

     1.   ONE
     2.   NEW LINE 2                     !CHANGE001
     3.   NEW LINE 3                     !CHANGE001
     4.   FOUR                           !**-2
     5.   FIVE
```

The values that you specified for position and length are stated in the header of the listing file.

16.1.5.2 **Using the /CHECKSUM Qualifier** - To obtain a checksum value, append the qualifier /CHECKSUM to the SLP initialization line. SLP processes the file, prints the checksum value in a message on your terminal, and exits.

If you want to check the accuracy of SLP editing commands, specify a checksum value using the form /CHECKSUM:n, where n is the checksum value previously calculated by SLP. If there is a mistake in the SLP command file (for example, if the edit command is -4,4 and you type -4,5), the checksum value will not match the value you specified with the /CHECKSUM qualifier. If the two values differ, SLP prints a diagnostic error message on your terminal, as described in Section 16.3.1.

SLP calculates a checksum value for all SLP edit commands except:

- The SLP initialization line
- Comments within the edit command line
- Spaces and/or tabs between characters included in the checksum calculation and those characters excluded from the calculation
- The second comma on an edit command line and anything following it (that is, audit trails and comments)
- The comment delimiter for an input line and any characters following it (the comment delimiter is defined as the first character in the current audit trail)

16.1.5.3 **Using the /NOOUTPUT and /LIST Qualifiers** - SLP processes input by sequence number. However, sequence numbers appear only in the listing file; they are not written to the output file.

To use SLP effectively, obtain an up-to-date numbered listing for use when you create the SLP command file. Numbered listings generated by other programs (such as the SOS editor and the VAX-11 MACRO assembler) will not necessarily be useful in preparing an SLP command file. Generate a SLP numbered listing by submitting an editing command in the following form:

```
     EDIT/SLP/OUTPUT=NL:/LIST[:list-file]  input-file
```

Here list-file is the name you optionally assign to the listing file that SLP produces, and input-file is the specification of the file whose lines are to be numbered. The slash (/) tells SLP to begin processing the files. SLP generates a numbered listing file, but does not produce an output file.

16.1.6 Specifying SLP Editing Commands

SLP editing commands let you update source files by adding, deleting, and replacing lines in a file. These commands contain certain characters that SLP interprets as operators. This section first describes these operators and the general form for specifying SLP editing commands. It then describes the editing commands used for specific editing functions.

16.1.6.1 SLP Operators - When SLP encounters any of the characters listed in Table 16-2 as the first character in an input line, it interprets the character as an operator.

Table 16-2: SLP Operators

Operator	Function
- (minus sign)	First character of an SLP editing command
\ (backslash)	Suppresses audit trail generation
% (percent sign)	Reenables audit trail generation
@ (at sign)	Invokes a further command file for SLP processing
/ (slash)	Terminates the editing session
< (less-than character)	Escape character

The percent sign (%) operator is used to reenable audit trail generation when generation has been suppressed by either the backslash (\) operator or the /NOAUDIT_TRAIL qualifier, described in Section 16.1.5.

The at sign (@) operator tells SLP to read further input from a another command file. This second command file can contain only SLP editing commands and new text lines.

The less-than character (<) operator is the escape character that lets you enter characters in the command file (in column 1) that SLP otherwise would interpret as operators. For example, </ hides the slash character from SLP, thereby enabling you to enter the slash into the output file without terminating the SLP editing session. You can use the less-than character as an escape character for all SLP operators listed in Table 16-2 (including itself).

16.1.6.2 General Form of an Editing Command

The general form of a SLP editing command is:

```
    -locator1[,locator2][,/audittrail/][;comment]
    inputline
          .
          .
          .
```

-

A minus-sign operator indicates that this is an SLP editing command line.

locator1

A line locator that causes SLP to move the current line pointer to a specified line. If only locator1 is specified, the current line pointer is moved to that line and SLP reads the next line in the editing command file. This field can be specified using any of the locator forms described below.

locator2

A line locator that defines a range of lines (that is, the range beginning with locator1 and ending with locator2) to be deleted or replaced. This field can be specified using any of the locator forms described below.

/audittrail/

A character string used to keep track of the update status of each line in the file. This audit trail is used to mark new or replaced lines in the file until the audit trail is either changed or suppressed. This argument must be delimited by slashes (/). If the editing command does not contain two locator fields, the audit trail specification must be preceded by two commas.

Audit trails generated by SLP use the first character of the specified string as a delimiter. Usually, the first character of the audit trail is set to match the comment delimiter of the source file being edited. Default audit trails are ;**NEW** for new lines and ;**-n for lines that indicate where text has been inserted.

inputline

A line of new text to be inserted into the file immediately following the current line. You can enter any number of input lines.

;comment

An optional comment. SLP ignores any text after a semicolon.

All fields in the command line are position-dependent; commas must be included as specified above.

The locator fields can take one of the following forms:

```
{ /string[...string]/ }
{       number        }  [+n]
{         .           }
```

Locator field parameters:

string

A string of ASCII characters. SLP locates the next line in which the string exists and moves the current line pointer to that line. If the locator is specified in the form /string...string/ (that is, two different strings of characters separated by three periods), SLP locates the line in which the first character string is followed by the second character string, regardless of what characters may be in between them.

number

A sequence number in the range of 1 through 9999 to which the current line pointer is to be moved.

n

A decimal value used as an offset from the line specified by the locator. Note that n is always preceded by a plus sign (+). You cannot back up from the locator.

A period represents the current line.

All forms of the line locator can be specified interchangeably in a command line.

SLP can only edit files sequentially. Once the current line pointer moves past a given line in the file, it cannot be returned to that line.

16.1.6.3 Adding Lines to a File - The SLP editing command for adding lines to a file contains only one locator field. Its form is:

```
-locator[,,/audittrail/][;comment]
```

The locator has one of the forms defined in Section 16.1.6.2.

If a numeric locator is specified, SLP inserts new line(s) after the line specified by sequence number. Any lines you enter are inserted as lines in the file.

If a string locator is specified, SLP locates the next occurrence of the string in the file and moves the current line pointer to the line containing the string. Any input lines following the command line are then added to the file.

If you specify an offset (+n), SLP moves the current line pointer n lines beyond the line specified in the locator field and then adds any new input lines to the file.

Because there is only one locator field, the audit trail specification must be preceded by two commas.

The example below shows how to add lines to a file. The input source file consists of the following lines:

```
     ABC
     DEF
     GHI
     KLM
     123456789
     456
     789
     CBA
     XYX
     987
```

The SLP command file consists of the following commands and text lines:

```
     $EDIT/SLP/LISTING/AUDIT_TRAIL:(POSITION:32) MYFILE.TST
     -/123/
     INSERT THIS LINE AFTER LINE 5
     /
```

SLP processing generates the following listing file:

```
MYFILE.TST,MYFILE=MYFILE.TST

      1.   ABC
      2.   DEF
      3.   GHI
      4.   KLM
      5.   123456789
      6.   INSERT THIS LINE AFTER LINE 5        ;**NEW**
      7.   456
      8.   789
      9.   CBA
     10.   XYX
     11.   987
```

SLP has applied sequence numbers to the lines and added an audit trail to the line following line 5, where SLP found the first occurrence of the string 123.

The next example uses the same input source file and the following new SLP command file:

```
     $EDIT/SLP/LISTING/AUDIT_TRAIL:(POSITION:32) MYFILE.TST
     -/DEF/+2
     THIS IS NEW TEXT
     /
```

SLP processing generates the following listing file:

```
MYFILE.TST,MYFILE - MYFILE.TST

      1.   ABC
      2.   DEF
      3.   GHI
      4.   KLM
      5.   THIS IS NEW TEXT                     ;**NEW**
      6.   123456789
      7.   456
      8.   789
      9.   CBA
     10.   XYX
     11.   987
```

Again, SLP has numbered the lines in sequence; this time the new input line is inserted two lines beyond the line containing the first occurrence of the string DEF.

16.1.6.4 Deleting Lines from a File - The SLP editing command for deleting lines from a file contains two locator fields. Its form is:

```
-locator1,locator2[,/audittrail/][;comment]
```

The locator1 and locator2 fields can take any of the forms described in Section 16.1.6.2. The first field, locator1, specifies the line where SLP is to begin deleting lines; locator2 specifies the last line to be deleted. SLP deletes all lines from locator1 through locator2, inclusive.

The example below shows how to delete lines from a file using SLP. The input source file consists of the following lines:

```
ABC
DEF
GHI
KLM
123456789
456
789
CBA
XYX
987
```

The SLP command file for this example is as follows:

```
$EDIT/SLP/LISTING/AUDIT_TRAIL:(POSITION:32) MYFILE.TST
-/1...9/,/XYX/
/
```

SLP processing generates the following listing file:

```
1.  ABC
2.  DEF
3.  GHI
4.  KLM
5.  987                            ;**-5
```

In this example, the ellipsis (...) is used to abbreviate the larger string 123456789. SLP searches for the first occurrence of the string 1 and the first occurrence of the string 9 on the line, assuming these two strings bracket a larger string, in this case, the string 123456789. SLP begins deleting lines at this line and continues deleting lines until it deletes the last line, specified by the string XYX. SLP applies the audit trail count of the lines it deleted to the next line in the output file.

Using the same input source file, this example shows how to delete a single line using the period locator. The command file for this example is as follows:

```
$EDIT/SLP/LISTING/AUDIT_TRAIL:(POSITION:32) MYFILE.TST
-/DEF/,.
/
```

SLP processing generates the following listing:

```
1.  ABC
2.  GHI                                    ;**-1
3.  KLM
4.  123456789
5.  456
6.  789
7.  CBA
8.  XYX
9.  987
```

SLP moves the current line pointer to the line containing the string DEF and then finds the period as the second locator field. Since the second locator field is specified, SLP interprets the editing command as a delete operation and deletes the line containing DEF.

16.1.6.5 Replacing Lines in a File - A replacement is a deletion followed by new text. The number of lines deleted need not match the number of lines added. To replace lines in a file, use the full 2-locator command form, as in the delete command. The first line locator field specifies the first line to be deleted. The second line locator field defines the last line in the range to be deleted, which, for replacement operations, is the line where new text is to be inserted.

For example, the command -4,.+4 instructs SLP to move the line pointer to line 4 and replace line 4 and the next four lines (as represented by .+4) with new input lines that immediately follow the command line. This command is equivalent to -4,8.

The example below shows how to delete lines from a file and replace them with new lines. The input source file consists of the following lines:

```
ABC
DEF
GHI
123456789
BCN
CRB
BUR
```

The SLP command file is as follows:

```
$EDIT/SLP/LISTING MYFILE.TST
-2,.+1
NEW LINE 2
NEW LINE 3
/
$EXIT
```

SLP processing generates the following listing file:

```
1.  ABC
2.  NEW LINE 2                            ;**NEW**
3.  NEW LINE 3                            ;**NEW**
4.  123456789                             ;**-2
5.  BCN
6.  CRB
7.  BUR
```

16.1.6.6 Specifying the Audit Trail Text - The following SLP edit command changes the text of the audit trail:

```
-,,/newtrail/
```

Here newtrail is the new value (text) of the audit trail. If the length of newtrail exceeds the length specified (or defaulted) for the /AUDIT_TRAIL qualifier, the audit trail is truncated to that length. (The default audit trail, ;**NEW**, is never truncated, even if you specify a length less that 8.)

All subsequent lines added will include the new audit trail text. All lines that indicate where lines have been deleted will include the first character of the new audit trail text as their first character. For example, if you specify the new audit trail JANUARY, the audit trail indicating a replaced line will be J**-2.

When you create a new audit trail, you may want to set the first character of the string to correspond to the comment delimiter that is used in the source file.

16.2 SUMSLP

SUMSLP is a batch-oriented editor similar to the SLP editor. It supplements the functions of SLP by allowing multiple command files to be applied to a single input file. The multiple command files are combined according to fixed rules.

16.2.1 Running SUMSLP

SUMSLP can be run either indirectly from a command procedure or interactively from your terminal. To invoke SUMSLP interactively, you issue a command line in response to the DCL prompt. To invoke SUMSLP from a command procedure, precede the command with a dollar sign ($). The command has the following format:

```
EDIT/SUM[/qualifier(s)]  input-file[/qualifier]
```

/qualifier(s)

A command or file qualifier, as described in Table 16-3. The /OUTPUT and /LIST qualifiers are command qualifiers only; the /UPDATE qualifier is a file qualifier only.

input-file

The file specification for the source file to be edited.

Table 16-3: SUMSLP Qualifiers

Qualifier	Meaning
/LIST[=file-spec]	Controls whether a sequence-numbered listing file, showing the original and inserted lines and an audit trail, is

(continued on next page)

Table 16-3 (Cont.): SUMSLP Qualifiers

Qualifier	Meaning
	produced during the editing process. If you do not specify a file, the listing file takes the same name as the input file, with a file type of LIS. You can specify another file type for the listing file, but LIS is the default. The listing file is described in Section 16.2.2.4.
/OUTPUT[=file-spec]	Specifies the output file to be used in the editing operation. If you do not specify a file, the output file has the same name and type as the input file, with a version number one higher than the highest existing version. The output file is described in Section 16.2.2.3.
/UPDATE[=(file-spec,...)]	Indicates the file or files containing the editing commands and changes to be applied to the input source file. If multiple file specifications are listed, they must be separated by commas, and the list must be enclosed in parentheses. The default file type of these files is initially UPD. Default values for the other elements of the file specification are initially taken from the input file specification; after the first file specification in a list, values default to those of the immediately preceding file specification. If no file specification or list of file specifications given, SUMSLP attempts to open a single update file with the same file name as the input file and a file type of UPD. If you do not include the /UPDATE qualifier in the command line, SUMSLP will not attempt to find an update file, but will generate any specified output or listing file. Enter the EDIT/SUM command with the /LIST qualifier but without the /UPDATE qualifier to generate a numbered listing of your source program.

Examples

1. $EDIT/SUM FILE1.MAR/UPDATE

   The input source file FILE1.MAR is updated with the SUMSLP command file FILE1.UPD.

2. `$EDIT/SUM    FILE2.MAR/UPDATE=UPD2`

   The input source file FILE2.MAR is updated with the SUMSLP command file UPD2.UPD.

3. `$EDIT/SUM     FILE3.MAR/UPDATE=(UPD3A,UPD3B.ENH,UPD3C)`

   The input source file FILE3.MAR is updated with the merged contents of SUMSLP command files UPD3A.UPD, UPD3B.ENH, and UPD3C.ENH. The editing commands in the three command files are applied according to the rules given in Section 16.2.3.

16.2.2 SUMSLP Input and Output Files

SUMSLP requires two types of input files: an input source file and one or more SUMSLP command files. SUMSLP produces two types of output files: a source output file and, if requested, a listing file. These four types of files are described in the following sections.

16.2.2.1 The Input Source File - The input source file is the file to be updated by SUMSLP. It can contain any number of lines.

16.2.2.2 The SUMSLP Command Procedure Files - SUMSLP command procedure files are very similar to SLP command procedure files, described in Section 16.1.2.2. They need not, however, include an initialization line.

The editing command lines in SUMSLP command files are identical to those used in SLP, with the following exceptions:

- The locator field, described in Section 16.1.6.2, cannot contain strings.
- Additional command files cannot be invoked with the at sign (@) operator.

As in SLP, the final editing command line must be followed by a line containing the slash operator (/), which serves as a terminator.

16.2.2.3 The Output File - The SUMSLP output file contains the input source file as updated by the additions and changes specified in the SUMSLP command file(s). It does not include an audit trail or line numbers.

If you do not include a file specification for the output file with the /OUTPUT qualifier in the EDIT/SUM command, the output file takes the same file name as the input source file, with a version number one higher than the existing version number.

16.2.2.4 The Listing File - The SUMSLP listing file is produced if you specify the /LIST qualifier in the EDIT/SUM command. If you do not specify another name for it, it takes the same file name as the input source file, with the file type of LIS. You can specify another file type, but LIS is the default.

The following example illustrates the generation of a listing file. The input source file, named MYFILE.TST, is:

```
ONE
TWO
THREE
FOUR
FIVE
SIX
SEVEN
EIGHT
NINE
TEN
```

There are two SUMSLP command files. The first, UPDATE.UPD, contains the following editing commands:

```
-3,3,/;21-MAR/
INSERTED LINE
/
```

The second SUMSLP command file, NEWLINES.UPD, contains the following editing commands:

```
-7,,/;22-MAR/
NEW LINE
/
```

When the commands in these SUMSLP command files are applied to the input source file, and the /LIST qualifier is applied, the following listing file is produced:

```
                           1 ONE
                           2 TWO
;21-MAR                   .1 INSERTED LINE
-1                         4 FOUR
                           5 FIVE
                           6 SIX
                           7 SEVEN
;22-MAR                   .1 NEW LINE
                           8 EIGHT
                           9 NINE
                          10 TEN
```

An audit trail is produced automatically unless it has been suppressed (see Section 16.1.5). This field will also contain a marker to indicate the number of lines deleted or replaced from the original file. The marker is placed on the first original line following a deletion and has the form -n, where n is the number of lines deleted.

The line numbers of inserted lines are distinguished from those of original lines by being preceded by a period. Inserted line numbers begin with .1 at the start of each group of new lines.

The source lines show the results of SUMSLP processing.

16.2.3 How SUMSLP Processes Files

SUMSLP applies the edits specified in the SUMSLP command file(s) to the source lines of the input source file. When a list of command files is specified with the /UPDATE qualifier, the editing commands in the various files are arranged according to the following rules:

1. The editing commands are merged into a single stream in ascending order according to the value of locator1 (as described in Section 16.1.6.2). All edits that do not overlap or conflict with any other edits are applied to the source file without any further processing.

2. Editing commands that do conflict are resolved according to the precedence of the SUMSLP command file in which the commands occur. Precedence of SUMSLP command files is determined by the position of the file specifications following /UPDATE. The file specification listed last after /UPDATE has the highest precedence.

   All inserts to the same source line are included in the output file; those from the SUMSLP command file with the highest precedence appear first.

   An operation that deletes or replaces a line will affect not only the specified line, but also any lower precedence inserts or replacements to the same line. A deletion that specifies a range of lines (for example, -10,15) will delete all lines occurring in that range, including inserted lines from SUMSLP command files of lower precedence.

16.3 SLP AND SUMSLP MESSAGES

The VAX/VMS System Messages and Recovery Procedures Manual lists the diagnostic messages issued by SLP and SUMSLP and provides explanations and suggested user actions for these messages.

CHAPTER 17

SYE UTILITY

The SYE Utility (SYE) is a system management tool that selectively reports the contents of an error log file.

The VAX/VMS system automatically writes messages to the latest version of an error log file named SYS$ERRORLOG:ERRLOG.SYS whenever one of the following events is detected:

- Errors -- Device errors, machine checks, bus errors, soft error correcting code (ECC) errors, asynchronous write errors, or hard ECC errors

- Configuration changes -- Volume mounts and dismounts

- System events -- Cold start-up, warm start-up, system failure start-up, message received from the Send Message to Error Logger ($SNDERR) system service, or a time stamp

All SYE reports are 72 columns wide, so they can be displayed at the terminal. Note that SYE reports are primarily intended to assist DIGITAL Field Service personnel. However, in some cases, they can assist in system management by identifying recurrent failures that indicate outside attention is required.

Additional details about error logging can be found in the VAX/VMS System Management and Operations Guide.

17.1 INVOKING AND TERMINATING SYE

The following command invokes the SYE Utility:

```
$ RUN SYS$SYSTEM:SYE
```

Only users with a system UIC or the SYSPRV privilege can access the error log file.

SYE normally runs to termination and issues a successful completion message on the device SYS$OUTPUT. To request additional SYE reports, you must reinvoke the utility. You can interrupt SYE at any time with CTRL/Y.

17.2 SYE PROMPTS

SYE displays a series of prompts to which you respond with a value or take the default by pressing CTRL/Z or the RETURN key. The prompt ends with a question mark and indicates the default within square brackets. The prompts and types of valid responses are shown below. The responses are described in the sections that follow.

```
                 Prompt                                  Valid Response

  _INPUT FILE  [SYS$ERRORLOG:ERRLOG.OLD]          ?   input-file-spec
  _OUTPUT FILE [SYS$OUTPUT]                       ?   output-file-spec
  _OPTIONS     [ROLL-UP]                          ?   report-option
  _DEVICE NAME [<CR>]                             ?   category-type
  _AFTER DATE  [FIRST ENTRY]                      ?   after-date-spec
  _BEFORE DATE [LAST ENTRY]                       ?   before-date-spec
```

17.3 VALID RESPONSES TO SYE PROMPTS

This section describes the responses you can enter to each of the SYE prompts in the sequence in which they should be entered.

17.3.1 Input-file-spec

The file specification for the input log file. If you do not specify a file, but simply respond by pressing RETURN, SYE uses the highest version of SYS$ERRORLOG:ERRLOG.OLD by default.

The file specification may be any valid VAX/VMS file specification. Wild card characters and logical names are allowed. You may also use the special file specification MAILBOX, which reports error log entries as they occur on the system, in the form you request. However, use of the MAILBOX feature of SYE requires the user privilege DIAGNOSE. The fields you omit default to those for VAX-11 FORTRAN unit 1, that is, the highest version of default disk:[default-directory]FOR001.DAT (SYE is written in FORTRAN).

17.3.2 Output-file-spec

The file specification for the output report file; the default is SYS$OUTPUT, which is usually your terminal.

If you want to direct the output to a specific file, simply identify it with any valid file specification. Logical names are allowed. Omitted fields default to those for VAX-11 FORTRAN unit 2, that is, the highest version of SYS$SYSROOT:[default-directory]FOR002.DAT.

17.3.3 Report-option

Any one of the following single-character abbreviations to request the type of report you want, or RETURN (to accept the default option, which is the roll-up report).

B (brief) -- a brief report of each error, configuration change, and system event being reported.

C (cryptic) -- a report of the contents of the device registers at the time a particular device error occurred. The values are represented in hexadecimal with no explanations. This report is only meaningful when errors on a single device are requested.

R (roll-up) -- a report providing totals for each category of error report requested (either a single category or all categories can be requested). This is the default.

S (standard) -- a full report of each error, configuration change, and system event being reported.

17.3.4 Category-type

Either the name of a device whose device errors you want reported or else a qualifier that requests particular categories of errors that you want to review. Details follow on how to specify these. If you respond by pressing RETURN, you receive a report of all errors (for every category and every device).

device name -- valid VAX/VMS-supported device names appear in the device name table in the VAX/VMS Command Language User's Guide. The asterisk (*) wild card character is allowed. For example, if you specify the device name DR*, you obtain a report of errors logged on all DR devices; if you specify DRA*, you obtain a report of errors on controller A of your DR device; a complete specification of DRA7 extracts errors only for the particular DR device on unit 7 of controller A. This is a change from earlier versions of SYE. Previously you could request reports on all DR devices by specifying DR. Now you must specify DR* to obtain the same report. You should not specify an underscore (_) with the device name; however, you can optionally specify a colon (:).

In addition, you can specify other DIGITAL-supported devices that may be on your system that were not provided with the VAX/VMS system.

SYE accepts a minus sign (-) as a prefix for a device name. That is, if you specify -DR*, the SYE report covers all devices except the DR disks.

category type qualifiers -- you can specify any one of the following category type qualifiers. Note that several of these qualifiers accept a device-name specification. For these qualifiers, you can specify one or more device-names, as described in the immediately preceding paragraphs. Specify multiple device-names in parentheses, separated by commas. Use a single asterisk (*) wild card character as the device-name specification to request a report of all the errors of that category on all the devices. (You can also use the asterisk wild card character in the device specification, as described above.)

/BU
All bugcheck entries.

/CO=(device-name[,...])
Configuration changes due to volume mounts and volume dismounts.

/CP
Machine checks, applicable bus errors, applicable interrupts, and uninitialized SCB vector interrupts (on a VAX-11/780 only).

/DA=(device-name[,...])
: Device attention entries. These device errors can occur even when there is no I/O request outstanding on the device. For example, such errors can occur when diagnostics are run on a TU78.

/DE=(device-name[,...])
: Device error bit(s) set entries.

/DT=(device-name[,...])
: Device I/O timeouts.

/ME
: Memory errors detected by the scanning code and interrupts and fatal memory errors logged by the machine check code.

/SY
: System events, such as system start ups, system powerfails, new error log file creation, system powerfail restart, time stamps, operator messages, network messages, and receipt of messages from the $SNDERR system service.

/UN
: Unknown device errors. This qualifier requests a report of errors found on all non-DIGITAL devices and all drivers not supported by DIGITAL. Furthermore, this specification covers DIGITAL-supported drivers that may have been released since the last release of SYE. When you request information on the unknown devices, the report is developed on a "best-try" basis, because SYE cannot fully interpret errors on these devices.

You can also specify a minus sign (-) as a prefix for any of the above qualifiers to indicate you want all the errors in every category except this one. However, you can not combine the minus sign on the qualifier with a minus sign on one or more of its device names.

17.3.5 After-date-spec

Time on error log after which reporting begins, in the following format:

```
dd-mmm-yyyy hh:mm:ss.cc
```

Note this is almost the format given in the VAX/VMS Command Language User's Guide for the absolute time format. The only difference is that you cannot specify a colon (:) between the year field (yyyy) and the hour field (hh). If you omit the entire time specification by pressing RETURN, SYE starts with the entries at the beginning of the log. You can omit portions of the time specification and obtain certain defaults. Refer to the description of the $BINTIM system service in the VAX/VMS System Service Reference Manual for the rules and examples of the defaults.

17.3.6 Before-date-spec

Time on error log before which reporting ends, in the following format:

```
dd-mmm-yyyy hh:mm:ss.cc
```

Note this is almost the format given in the VAX/VMS Command Language User's Guide for the absolute time format. The only difference is that you cannot specify a colon (:) between the year field (yyyy) and the hour field (hh). If you omit the entire time specification by pressing RETURN, SYE includes all entries until the end of the log is reached. You can omit portions of the time specification and obtain certain defaults. Refer to the description of the $BINTIM system service in the VAX/VMS System Service Reference Manual for the rules and examples of the defaults.

17.4 ERROR MESSAGES

SYE rarely issues error messages on incorrect input; ususally it simply reprompts. A few error messages originated by SYE have a facility code of SYE or some other system component such as RMS or SYSTEM. The VAX/VMS System Messages and Recovery Procedures Manual lists these messages and provides explanations and suggested user actions. If any other type of error message appears in the report, you should rerun SYE to eliminate the error. Most SYE error messages are reported as VAX-11 FORTRAN messages. If you receive a recurring error message that does not disappear when you rerun SYE, you should submit a Software Performance Report (SPR) to DIGITAL. (SPRs are described in the VAX/VMS System Management and Operations Guide.)

CHAPTER 18

SYSTEM GENERATION UTILITY (SYSGEN)

The System Generation Utility (SYSGEN) is a system management tool that performs system generation. This chapter describes how to invoke SYSGEN and how to use its commands and displays. Additional information on the system parameters that SYSGEN controls and their impact on system operation appears in the VAX/VMS System Management and Operations Guide.

Table 18-1 summarizes the SYSGEN commands by format and function.

Table 18-1: SYSGEN Command Summary

Format	Function
AUTOCONFIGURE nexus [/EXCLUDE=(device-name[,...])] ALL [/LOG] [/SELECT=(device-name[,...])]	Automatically connects the devices physically attached to the system and loads their drivers
CONFIGURE [/INPUT=file-spec] [/OUTPUT=file-spec] [/[NO]RESET]	Requests the UNIBUS device names and outputs the set of CSR and vector addresses that AUTOCONFIGURE uses
CONNECT device /ADAPTER=nexus [/CSR=csr-addr] [/DRIVERNAME=driver] [/MAXUNITS=max-unit-cnt] [/NUMVEC=vector-cnt] [/VECTOR=vector-addr]	Connects hardware devices and loads their drivers (if they are not already loaded)
CONNECT device /NOADAPTER [/DRIVERNAME=driver]	Connects software devices and loads their drivers (if they are not already loaded)
CONNECT CONSOLE	Connects the console block storage device and loads its driver
CREATE file-spec /SIZE=block-count [/[NO]CONTIGUOUS]	Creates or extends a paging, swapping, or dump file
DISABLE CHECKS	Disables range checks

(continued on next page)

Table 18-1 (Cont.): SYSGEN Command Summary

Format	Function
ENABLE CHECKS	Enables range checks
EXIT	Terminates SYSGEN
HELP [command-name [/qualifier]]	Provides information on the SYSGEN commands
INSTALL file-spec /PAGEFILE /SWAPFILE	Activates a secondary paging or swapping file
LOAD file-spec	Loads a device driver
RELOAD file-spec	Loads a new version of an existing device driver
SET parameter-name value	Modifies the value of a system generation parameter in the SYSGEN work area
SET /OUTPUT [=] file-spec	Defines an output file for the session
SET /STARTUP file-spec	Names the current site-independent start-up command procedure
SHARE MPMn mpm-name [/MAXCEFCLUSTERS=max-cef] [/MAXGBLSECTIONS=max-gbl] [/MAXMAILBOXES=max-mail]	Connects a multiport memory unit
SHARE MPMn mpm-name /INITIALIZE [/CEFCLUSTERS=cef] [/GBLSECTIONS=gbl] [/MAILBOXES=mail] [/MAXCEFCLUSTERS=max-cef] [/MAXGBLSECTIONS=max-gbl] [/MAXMAILBOXES=max-mail] [/POOLBCOUNT=block-cnt] [/POOLBSIZE=block-size] [/PRQCOUNT=prq-cnt]	Initializes a multiport memory unit
SHOW {parameter-name /ACP /ALL /DYNAMIC /GEN /JOB /MAJOR /NAMES /PQL /RMS /SCS /SPECIAL /SYS /TTY} [/HEX]	Displays the values of system parameters in the SYSGEN work area, plus the default, minimum, and maximum values of the parameters and their units of measure

(continued on next page)

Table 18-1 (Cont.): SYSGEN Command Summary

Format	Function
SHOW /ADAPTER	Lists all the nexus numbers and generic names
SHOW /CONFIGURATION [/ADAPTER=nexus] [/COMMAND_FILE] [/OUTPUT[=file-spec]]	Shows devices by name, number of units, nexus number, and adapter type, as well as by CSR and vector addresses
SHOW /DEVICE[=driver-name] /DRIVER[=driver-name]	Displays information on connected devices and loaded drivers
SHOW /STARTUP	Displays the name of the current site-independent start-up command procedure
SHOW /UNIBUS [/ADAPTER=nexus]	Displays the addresses in UNIBUS I/O space that can be addressed
USE file-spec CURRENT ACTIVE DEFAULT	Initializes the SYSGEN work area with system parameter values from a parameter file, the current system image, the active system, or the default list
WRITE file-spec CURRENT ACTIVE	Writes the system parameter values from the SYSGEN work area to a parameter file, the current system image, or the active system

Many of the SYSGEN commands are specific to particular topics or functions:

- System parameters -- USE, SET, ENABLE, DISABLE, WRITE, and SHOW
- Devices and device drivers -- AUTOCONFIGURE, CONFIGURE, CONNECT, LOAD, RELOAD, SHOW/CONFIGURATION, SHOW/DEVICE, SHOW/DRIVER, and SHOW/UNIBUS
- System files -- CREATE and INSTALL
- Start-up command procedure -- SET/STARTUP and SHOW/STARTUP
- Multiport memory -- SHARE

A subset of the SYSGEN commands can be used during bootstrap operations; see the software installation guide for your VAX-11 processor for more information.

18.1 INVOKING AND TERMINATING SYSGEN

The following command invokes the utility:

```
$ RUN SYS$SYSTEM:SYSGEN
```

The system responds with the following prompt:

```
SYSGEN>
```

You can then enter any of the commands listed in Table 18-1. These commands follow the standard rules of grammar as specified in the VAX/VMS Command Language User's Guide.

You can terminate SYSGEN in a normal way with the SYSGEN command EXIT. SYSGEN also terminates if you press CTRL/Z.

18.2 SYSGEN COMMANDS

The following sections describe the individual SYSGEN commands in alphabetical order.

18.2.1 AUTOCONFIGURE Command

The AUTOCONFIGURE command automatically connects devices that are physically attached to the system and loads their drivers. It has the following format:

```
AUTOCONFIGURE  nexus    [/EXCLUDE=(device-name[,...])]
                ALL     [/LOG]
                        [/SELECT=(device-name[,...])]
```

nexus
Nexus (backplane interconnect arbitration line) or slot number of the single UNIBUS or MASSBUS adapter that is to be configured, or the keyword ALL. The nexus can be expressed as an integer or with one of the names listed by the SYSGEN command SHOW/ADAPTER.

device-name
Standard device name as shown in the device names table in the VAX/VMS Command Language User's Guide. You may (optionally) include a controller designation, but do not include a unit number. If the controller designation is omitted, device-name refers to all devices of the specified type; defaults to all devices on the adapter. Parentheses can be omitted for a single device. With the /SELECT qualifier, you identify the particular device types desired; with the /EXCLUDE qualifier, you identify only those device types not desired for autoconfiguration. Note that the /SELECT and /EXCLUDE qualifiers should not be used in combination, that is, they are mutually exclusive.

The autoconfigure operation assigns standard device names and assumes standard driver names.

The optional /LOG qualifier produces a display on the SYS$OUTPUT device of the controller and its units after they have been successfully autoconfigured. In other words, each controller and its associated units are displayed only after AUTOCONFIGURE has found the next controller. Thus, the error message displays precede the display of the controller and units that caused the error.

The following example automatically configures all standard devices:

```
     SYSGEN>  AUTOCONFIGURE ALL
```

The next example automatically configures all terminals, all magnetic tape units on controller A, and all line printers:

```
     SYSGEN>  AUTOCONFIGURE ALL /SELECT=(TT,MTA,LP)
```

The following example illustrates the use of the /EXCLUDE qualifier to autoconfigure all but the DMC11 devices, assuming DECnet-VAX will not be operating on this system.

```
     SYSGEN>  AUTOCONFIGURE ALL /EXCLUDE=XM
```

Use of the AUTOCONFIGURE command requires the CMKRNL privilege.

18.2.2 CONFIGURE Command

The CONFIGURE command requests UNIBUS device names and outputs the set of CSR and vector addresses that AUTOCONFIGURE will use. It has the following format:

```
     CONFIGURE  [/INPUT=file-spec]
            [/OUTPUT=file-spec]
            [/[NO]RESET]
```

file-spec

Name of the file for input or output. You can specify /INPUT to read the data from a previously prepared input file. If you omit the /INPUT qualifier, by default input is read from the SYS$INPUT device.

You can save the output from CONFIGURE by specifying an output file with /OUTPUT. By default, the output is directed to the SYS$OUTPUT device. The default output file type is LIS.

The /NORESET qualifier is useful with multiple UNIBUS systems. When you specify /NORESET, it is not necessary to specify the second parameter (p) on subsequent CONFIGURE commands, since the controller names are not reset. By default, if you omit /NORESET, the controller names are reset.

CONFIGURE prompts with:

```
     DEVICE>
```

Input should be in the form:

```
     device[[,n],p]
```

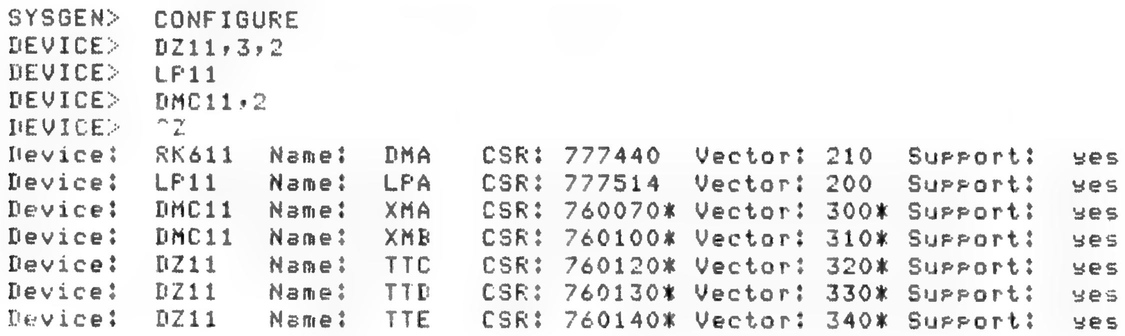
device

Name of the controller; see the section, The SYSGEN Device Table, in Chapter 14 of the VAX/VMS Guide to Writing a Device Driver. The controller names displayed there in the "Name" column are those accepted by the CONFIGURE command. You cannot abbreviate controller names.

You can optionally specify n, the number of devices on the UNIBUS being configured and p, the optional number of devices on all previous UNIBUSes in a multiple UNIBUS system. Note that p does not affect the addresses generated, only the device names. By default, n is 1 and p is 0.

SYSGEN continues to prompt for devices until you respond with CTRL/Z. It then outputs the CSR and vector addresses.

Figure 18-1 illustrates the use of the CONFIGURE command to calculate the UNIBUS CSR and vector addresses. The support field in the display indicates whether DIGITAL provides a supported driver for this device with VAX/VMS.



```
SYSGEN>  CONFIGURE
DEVICE>  DZ11,3,2
DEVICE>  LP11
DEVICE>  DMC11,2
DEVICE>  ^Z
Device:  RK611   Name:  DMA   CSR: 777440  Vector: 210  Support:  yes
Device:  LP11    Name:  LPA   CSR: 777514  Vector: 200  Support:  yes
Device:  DMC11   Name:  XMA   CSR: 760070* Vector: 300* Support:  yes
Device:  DMC11   Name:  XMB   CSR: 760100* Vector: 310* Support:  yes
Device:  DZ11    Name:  TTC   CSR: 760120* Vector: 320* Support:  yes
Device:  DZ11    Name:  TTD   CSR: 760130* Vector: 330* Support:  yes
Device:  DZ11    Name:  TTE   CSR: 760140* Vector: 340* Support:  yes
```

* indicates a floating address

Figure 18-1: SYSGEN Command CONFIGURE Example

18.2.3 CONNECT (Hardware) Command

The CONNECT (hardware) command connects a hardware device and loads its driver, if the driver is not already loaded. It has the following format:

```
CONNECT device /ADAPTER=nexus
                [/CSR=csr-addr]
                [/DRIVERNAME=driver]
                [/MAXUNITS=max-unit-cnt]
                [/NUMVEC=vector-cnt]
                [/VECTOR=vector-addr]
```

device
: Name of the device; should take the form device type, controller, unit -- for example, LPA0 for unit 0 on controller A of device type LP.

nexus
: Nexus number of the UNIBUS or MASSBUS adapter to which the device is attached. The nexus can be expressed as an integer or as one of the names listed by the SYSGEN command SHOW/ADAPTER.

csr-addr
: UNIBUS address of the controller status register for the device; must be specified for UNIBUS devices.

driver
: Name of the driver as recorded in the prologue table; if the driver has not been loaded, the system assumes that the driver name is also the name of an executable image (file type of EXE) in SYS$SYSTEM, and loads the driver; defaults to the first two characters of the device name plus DRIVER.

max-unit-cnt
Maximum number of units the controller can support (that is, the number of UCB slots in the IDB); defaults to the number specified in the prologue table of the driver, or to 8 if the number is not specified in the prologue table.

vector-cnt
Number of interrupt vectors for the device; defaults to 1.

vector-addr
UNIBUS address of the interrupt vector for the device, or the lowest vector if there is more than one. Must be specified for UNIBUS devices.

You normally use this command to connect devices that are not physically attached to the system or whose drivers are not standard. The following example connects the device named LPA0 to the driver named LPDRIVER (the default driver name), and loads the driver if it is not already loaded:

```
SYSGEN>  CONNECT LPA0 /ADAPTER=3/CSR=%O777514/VECTOR=%O200
```

The next example performs the same operation, but uses a nonstandard driver named LP2DRIVER:

```
SYSGEN>  CONNECT LPA0 /ADAPTER=3/CSR=%O777514-
SYSGEN>  /DRIVERNAME=LP2DRIVER/VECTOR=%O200
```

CAUTION

You should exercise extreme caution when using the CONNECT (hardware) command, because the system does little error checking. An incorrect vector address or misspelled device name, for example, will damage the I/O data base and very likely cause the system to fail.

The *VAX/VMS Guide to Writing a Device Driver* contains more detailed information on loading device drivers and connecting devices.

Use of the CONNECT (hardware) command requires the CMKRNL privilege.

18.2.4 CONNECT (Software) Command

The CONNECT (software) command connects a software device and loads its driver if it is not already loaded. It has the following format:

```
CONNECT device /NOADAPTER
   [/DRIVERNAME=driver]
```

device
Name of the device.

driver
Name of the driver as recorded in the prologue table; if the driver has not been loaded, the system assumes that the driver name is also the name of an executable (file type of EXE) image in SYS$SYSTEM, and loads the driver. Defaults to the first two characters of the device name plus DRIVER.

The driver prologue table must specify ADAPTER=NULL for this command to work.

The following example connects the device NET to the driver NETDRIVER, and loads the driver if it is not already loaded:

```
     SYSGEN>  CONNECT NET /NOADAPTER/DRIVER=NETDRIVER
```

Use of the CONNECT (software) command requires the CMKRNL privilege.

18.2.5 CONNECT (Console) Command

The CONNECT (console) command connects the console block storage device(s) and loads the driver. It has the following format:

```
     CONNECT CONSOLE
```

The console block storage device driver is CSDRIVER.EXE.

Use of the CONNECT (console) command requires the CMKRNL privilege.

18.2.6 CREATE Command

The CREATE command can be used to create or extend a file that can be used as a paging, swapping, or dump file. It has the following format:

```
     CREATE file-spec /SIZE=block-count [/[NO]CONTIGUOUS]
```

file-spec
: Name of the file. The default file type is SYS. Primary paging and swapping files have the names SYS$SYSTEM:PAGEFILE.SYS and SYS$SYSTEM:SWAPFILE.SYS. A secondary paging or swapping file only takes effect after you install it with the SYSGEN commmand INSTALL (see Section 18.2.11).

  The dump file name is SYS$SYSTEM:SYSDUMP.DMP. When you create a new SYSDUMP.DMP file, you must explicitly specify the file type DMP. The only way you make a new SYSDUMP.DMP file active is by rebooting the system.

block-count
: Number of blocks to be allocated to the file when the operation is complete.

The default is /NOCONTIGUOUS, which implies a contiguous-best-try file. If /NOCONTIGUOUS is specified, the following logic is used:

1. If the file does not exist, SYSGEN creates it.

2. If the file does exist and the size specified by the /SIZE qualifier is smaller than the current size, then SYSGEN creates a new file of the new size.

3. If the file does exist and the size specified by the /SIZE qualifier is larger than the current size, then SYSGEN extends the current file to the new size.

The /CONTIGUOUS qualifier forces the creation of a new file and guarantees that the file will be contiguous.

The following example creates a contiguous paging file of 95,000 blocks:

```
SYSGEN>  CREATE SYS$SYSTEM:PAGEFILE /SIZE=95000 /CONTIGUOUS
%SYSGEN-I-CREATED, SYS$SYSROOT:[SYSEXE]PAGEFILE.SYS;2 created
```

The following example then extends the same file to 100000 blocks:

```
SYSGEN>  CREATE SYS$SYSTEM:PAGEFILE /SIZE=100000
%SYSGEN-I-EXTENDED, SYS$SYSROOT:[SYSEXE]PAGEFILE.SYS;2 extended
```

You do not have to issue this command directly for the creation of primary files because the installation of VAX/VMS creates them. However, you should execute the command procedure SYS$UPDATE:SWAPFILES to change the size of the primary files.

Files created with SYSGEN receive the default protection of the process that invoked SYSGEN. Thus, DIGITAL recommends that you use the SWAPFILES command procedure, which includes a SET PROTECTION command to establish appropriate protection for the files after it creates them with SYSGEN.

18.2.7 DISABLE Command

The DISABLE command inhibits range checks on parameter values specified in SET commands. It has the following format:

```
DISABLE CHECKS
```

Initially, checks are enabled. If you specify a value above the maximum, the system sets the parameter to the maximum and issues an error message. If you specify a value below the minimum, the system sets the parameter to the minimum and issues an error message. In the following example, the initial attempt to set WSMAX below the minimum fails because range checks are enabled. However, once the user disables range checks, the SET WSMAX command succeeds.

```
SYSGEN>  SET WSMAX 20
%SYSGEN-W-SETMIN, Value set to minimum for parameter WSMAX
SYSGEN>  DISABLE CHECKS
SYSGEN>  SET WSMAX 20
```

WARNING

> The minimum and maximum values established by DIGITAL are intended as fairly strict guidelines. Setting parameter values outside these limits can result in system failures or hangs. Thus, the use of DISABLE CHECKS is not recommended.

18.2.8 ENABLE Command

The ENABLE command ensures that range checks are in effect. It has the following format:

```
ENABLE CHECKS
```

Initially, range checks are enabled, so that this command need be used only after a DISABLE command was issued earlier in the session.

18.2.9 EXIT Command

The EXIT command returns you to DCL command level. It has the following format:

```
EXIT
```

You can also return to DCL command level by pressing CTRL/Z.

18.2.10 HELP Command

The HELP command lists and explains the SYSGEN commands. It has the following format:

```
HELP [command-name [/qualifier]]
```

command-name
Name of a SYSGEN command or the keyword PARAMETERS. The command HELP PARAMETERS displays the system parameters and a brief description of each.

/qualifier
Name of a SYSGEN command qualifier.

If you do not specify either a command name or the keyword PARAMETERS, HELP displays general information on the commands for which help is available. It then prompts with "Topic?". You can supply a command name, the keyword PARAMETERS, or press RETURN. Specification of command names and qualifiers obtains more detailed information. If you respond with a RETURN, the HELP command exits. You can also exit from the HELP command by pressing CTRL/Z. Do not respond with CTRL/Y unless you want to exit from both HELP and SYSGEN.

If the command you seek help on accepts qualifiers, the display of information for the command is followed by a prompt that includes the command name and "Subtopic?". You can respond to this prompt with a qualifier name or press RETURN. If you respond with a RETURN, HELP prompts with "Topic?". If you want to exit the HELP command directly from this level, press CTRL/Z.

18.2.11 INSTALL Command

The INSTALL command activates a secondary paging or swapping file. It has the following format:

```
INSTALL file-spec  /PAGEFILE
                   /SWAPFILE
```

file-spec
Name of a secondary paging or swapping file created with the SYSGEN command CREATE; the file type defaults to SYS.

Either /PAGEFILE or /SWAPFILE must be specified to indicate the type of file being activated.

The following example installs a secondary paging file:

```
SYSGEN>  INSTALL SYS$SYSTEM:PAGEFILE.SYS /PAGEFILE
```

The new paging or swapping file is effective until system shutdown.

Use of the INSTALL command requires the CMKRNL privilege.

18.2.12 LOAD Command

The LOAD command loads an I/O driver. It has the following format:

```
LOAD file-spec
```

file-spec
: File specification of the driver image; the file type defaults to EXE.

If the entire file specification is the same as that of a driver already loaded, no load takes place. If only the file name is the same as that of a driver that is already loaded (but the file specification as a whole is different), the specified driver replaces the existing driver.

DIGITAL supplies drivers for the standard devices.

The following example loads the standard driver for a remote terminal:

```
SYSGEN>  LOAD SYS$SYSTEM:RTTDRIVER
```

The *VAX/VMS Guide to Writing a Device Driver* contains more detailed information on loading a driver.

Use of the LOAD command requires the CMKRNL privilege.

18.2.13 RELOAD Command

The RELOAD command replaces a loaded device driver with a new version. It has the following format:

```
RELOAD file-spec
```

file-spec
: File specification of the new driver image; the file type defaults to EXE.

The specified image is loaded and replaces any existing driver with the same file specification. The following example reloads the remote terminal driver:

```
SYSGEN>  RELOAD SYS$SYSTEM:RTTDRIVER
```

Use of the RELOAD command requires the CMKRNL privilege.

18.2.14 SET (Parameter) Command

The SET (parameter) command assigns a value to a system parameter as it exists in the SYSGEN work area. The command has the following format:

```
SET parameter-name value
```

parameter-name
: Name of a system parameter (as specified in the chapter, System Parameters, in the *VAX/VMS System Management and Operations Guide*), or a period (.). A period is interpreted as a request for the system parameter specified in the last SET (parameter) or SHOW (parameter) command.

You can display the system parameters and request information on them with the SYSGEN command HELP PARAMETERS.

value
: An integer or the keyword DEFAULT; integer values must be within the defined minimum and maximum values for the parameter unless the SYSGEN command DISABLE CHECKS was specified; DEFAULT means the default value for the parameter. You can display the maximum, minimum, and default values for any parameter with the SYSGEN command SHOW.

This command does not modify parameter files, the current system image on disk, or the active system; for information on performing these modifications, see the WRITE command below.

The following example assigns a value of 20 to the PFCDEFAULT parameter:

```
SYSGEN>  SET PFCDEFAULT 20
```

The next example assigns the default value (40) to the GBLSECTIONS parameter:

```
SYSGEN>  SET GBLSECTIONS DEFAULT
```

See the description of the SHOW (parameter) command for an illustration of the use of the period in place of a parameter name.

18.2.15 SET (Output) Command

The SET (output) command establishes a file to be used for output during the session. By default the output device is SYS$OUTPUT, but you can use the SET (output) command to designate an alternate device. The SET (output) command has the following format:

```
SET /OUTPUT [=] file-spec
```

file-spec
: File specification of an output file for recording the session. The default file type is LIS.

At any time you can direct the output back to the default SYS$OUTPUT device by using the SET/OUTPUT=SYS$OUTPUT command.

In the following example, output is directed to the file PARAMS.LIS to capture a complete list of all the system parameters and their values.

```
SYSGEN>  SET/OUTPUT=PARAMS.LIS
SYSGEN>  SHOW/ALL
SYSGEN>  SHOW/SPECIAL
SYSGEN>  EXIT
```

18.2.16 SET (Start-Up Command Procedure) Command

The SET (start-up command procedure) command names the site-independent start-up command procedure to be associated with a parameter file for subsequent bootstrap operations. It has the following format:

```
SET /STARTUP file-spec
```

file-spec
  File specification of a start-up command procedure on the system disk (maximum of 31 characters).

The SET command does not modify either parameter files or the current system image on disk; for information on performing these modifications, see the WRITE command below.

The following example assigns SYS$SYSTEM:XSTARTUP.COM as the current site-independent start-up command procedure:

```
     SYSGEN>  SET/STARTUP SYS$SYSTEM:XSTARTUP.COM
```

The initial site-independent start-up command procedure (as named in the software distribution kit) is SYS$SYSTEM:STARTUP.COM.

18.2.17 SHARE (Connect) Command

The SHARE (connect) command connects a processor to a multiport memory unit already initialized by this or another processor. The command has the following format:

```
     SHARE MPMn mpm-name [/MAXCEFCLUSTERS=max-cef]
                         [/MAXGBLSECTIONS=max-gbl]
                         [/MAXMAILBOXES=max-mail]
```

n
  Number on the front panel of the multiport memory unit being connected.

mpm-name
  Name of the multiport memory unit as specified in a previous SHARE (initialize) command.

max-cef
  Maximum common event flag clusters that this processor can create in the multiport memory unit; defaults to no limit.

max-gbl
  Maximum global sections that this processor can create in the multiport memory unit; defaults to no limit.

max-mail
  Maximum mailboxes that this processor can create in the multiport memory unit; defaults to no limit.

The number and name of the multiport memory unit must match those of an initialized unit or an error condition results.

The following example connects a multiport memory unit. Since no qualifiers are specified, defaults apply to all the parameters.

```
     SYSGEN>  SHARE MPM1 SHR_MEM_1
```

The unit with a 1 on the front panel must be initialized with the name SHR_MEM_1 for the command to work.

Use of the SHARE (connect) command requires the CMKRNL privilege.

18.2.18 SHARE (Initialize) Command

The SHARE (initialize) command initializes a multiport memory unit, if it is not already initialized, and connects it to the processor on which SYSGEN is running. The command has the following format:

```
     SHARE MPMn mpm-name /INITIALIZE
                         [/CEFCLUSTERS=cef]
                         [/GBLSECTIONS=gbl]
                         [/MAILBOXES=mail]
                         [/MAXCEFCLUSTERS=max-cef]
                         [/MAXGBLSECTIONS=max-gbl]
                         [/MAXMAILBOXES=max-mail]
                         [/POOLBCOUNT=block-cnt]
                         [/POOLBSIZE=block-size]
                         [/PRQCOUNT=prq-cnt]
```

n

Number on the front panel of the multiport memory unit being initialized.

mpm-name

Name by which the multiport memory unit is to be known to systems using it; consists of 1 through 15 alphanumeric characters, including dollar signs ($) and underscore characters (_).

cef

Total common event flag clusters permitted in the multiport memory unit; defaults to 32.

gbl

Total global sections permitted in the multiport memory unit; defaults to 32.

mail

Total mailboxes permitted in the multiport memory unit; defaults to 32.

max-cef

Maximum common event flag clusters that this processor can create in the multiport memory unit; defaults to no limit.

max-gbl

Maximum global sections that this processor can create in the multiport memory unit; defaults to no limit.

max-mail

Maximum mailboxes that this processor can create in the multiport memory unit; defaults to no limit.

block-cnt

Number of blocks allocated to the multiport memory unit's dynamic pool; defaults to 128 blocks.

block-size

Size of each block in the dynamic pool; defaults to 128 bytes.

prq-cnt

Number of interprocessor request blocks (PRQs) allocated; defaults to 64 blocks.

The dynamic pool is used for AST control blocks, message buffers, and other small dynamic structures.

If the specified multiport memory unit is already initialized and connected to other active processors, the gbl, mail, cef, block-cnt, block-size, and prq-cnt parameter values are ignored, and the unit is simply connected to the processor.

The following example initializes a multiport memory unit with defaults on all but the gbl, mail, and cef parameters:

```
     SYSGEN>  SHARE MPM1 SHR_MEM_1 /INITIALIZE -
     SYSGEN>  /GBLSECTIONS=128/MAILBOXES=64/CEFCLUSTERS=0
```

The number of the multiport memory unit as it appears on the front panel is 1. You name the unit SHR_MEM_1.

Use of the SHARE (initialize) command requires the CMKRNL privilege.

18.2.19 SHOW (Parameter) Command

The SHOW (parameter) command displays the values of system parameters in the SYSGEN work area, plus the default, minimum, and maximum values of the parameters, and their units of measure. It has the following format:

```
           / parameter-name                     \
          |  /ACP                                |
          |  /ALL                                |
          |  /DYNAMIC                            |
          |  /GEN                                |
          |  /JOB                                |
          |  /MAJOR                              |
     SHOW <  /NAMES                  [/HEX]      >
          |  /PQL                                |
          |  /RMS                                |
          |  /SCS                                |
          |  /SPECIAL                            |
          |  /SYS                                |
           \ /TTY                               /
```

parameter-name
    Name of a system parameter as specified in the chapter, System Parameters, in the VAX/VMS System Management and Operations Guide or a period (.). The period is interpreted as a request for the system parameter specified in the last SET (parameter) or SHOW (parameter) command.

You can specify only one parameter name or one of the parameter type qualifiers. Specification of a parameter name displays the values of that parameter. Specification of a parameter type qualifier displays the values of all system parameters of its type as follows: ACP parameters (/ACP), all parameters except the SPECIAL parameters (/ALL), all DYNAMIC parameters (/DYNAMIC), all GEN parameters (/GEN), all JOB parameters (/JOB), all MAJOR parameters (/MAJOR), all PQL parameters (/PQL), all VAX-11 RMS parameters (/RMS), all SCS parameters (/SCS), all SPECIAL parameters reserved for DIGITAL's use (/SPECIAL), all SYS parameters (/SYS), all terminal parameters (/TTY), or just the names of all parameters (/NAMES).

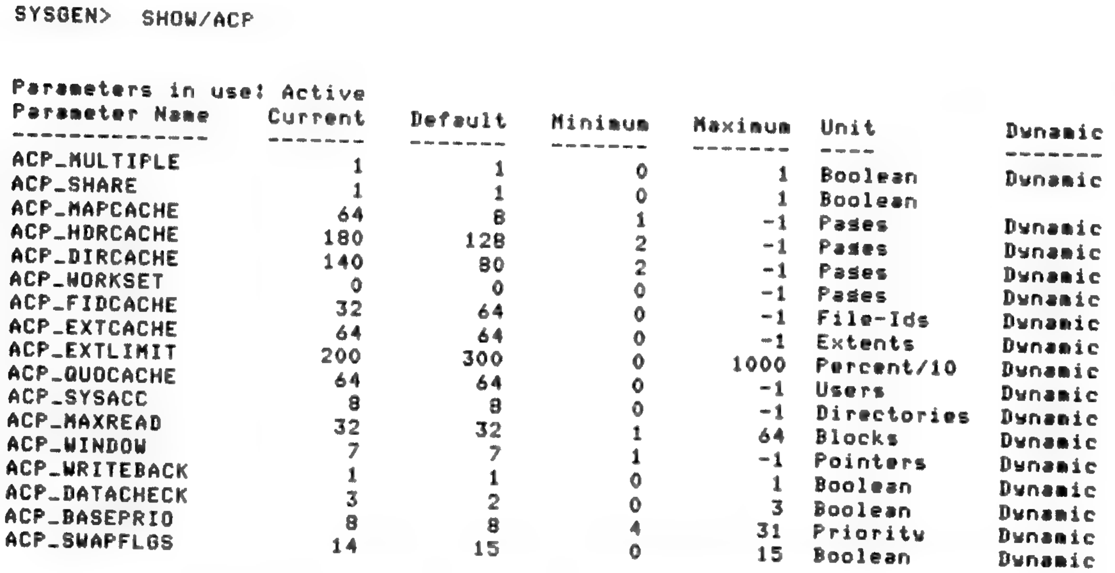
The display of the parameter values is in decimal. You can optionally request the values of the parameter(s) be displayed in hexadecimal representation with the /HEX qualifier after the system parameter name or the parameter type. If you specify /HEX with the /NAMES qualifier, /HEX is ignored.

> **NOTE**
>
> When parameter names are abbreviated, the first parameter matching the abbreviation is selected for display. No ambiguity checks are made. For example, a specification of SHOW GBL displays the GBLSECTIONS parameter. To display the GBLPAGFIL parameter you must specify SHOW GBLPAGF (to avoid further ambiguity with the GBLPAGES parameter).

You can use a period (.) to indicate that you want to work with the system parameter that was specified in the last SET (parameter) or SHOW (parameter) command, as in the example below. Here the user first displays the values of the GBLSECTIONS parameter, then refers to the parameter with a period to set its current value to 110. The next SHOW command also uses the period notation to obtain confirmation that the change occurred.

```
SYSGEN>  SHOW GBLSECTIONS
  GBLSECTIONS                100          40          20          -1  Sections
SYSGEN>  SET . 110
SYSGEN>  SHOW .
  GBLSECTIONS                110          40          20          -1  Sections
```

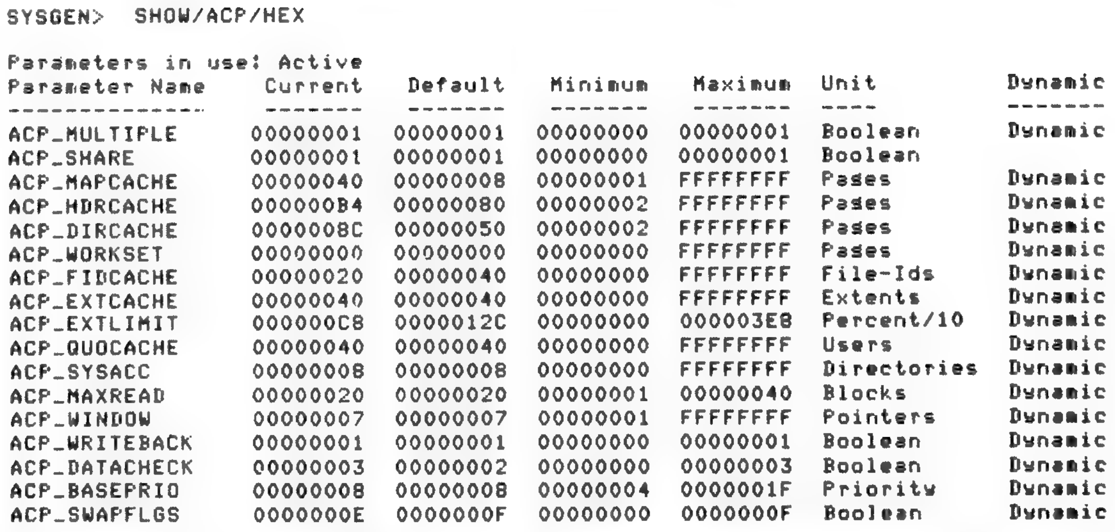
Figure 18-2 illustrates the output produced by the SYSGEN command SHOW/ACP.



```
SYSGEN>  SHOW/ACP


Parameters in use: Active
Parameter Name     Current   Default   Minimum   Maximum  Unit          Dynamic
--------------     -------   -------   -------   -------  ----          -------
ACP_MULTIPLE             1         1         0         1  Boolean       Dynamic
ACP_SHARE                1         1         0         1  Boolean
ACP_MAPCACHE            64         8         1        -1  Pages         Dynamic
ACP_HDRCACHE           180       128         2        -1  Pages         Dynamic
ACP_DIRCACHE           140        80         2        -1  Pages         Dynamic
ACP_WORKSET              0         0         0        -1  Pages         Dynamic
ACP_FIDCACHE            32        64         0        -1  File-Ids      Dynamic
ACP_EXTCACHE            64        64         0        -1  Extents       Dynamic
ACP_EXTLIMIT           200       300         0      1000  Percent/10    Dynamic
ACP_QUOCACHE            64        64         0        -1  Users         Dynamic
ACP_SYSACC               8         8         0        -1  Directories   Dynamic
ACP_MAXREAD             32        32         1        64  Blocks        Dynamic
ACP_WINDOW               7         7         1        -1  Pointers      Dynamic
ACP_WRITEBACK            1         1         0         1  Boolean       Dynamic
ACP_DATACHECK            3         2         0         3  Boolean       Dynamic
ACP_BASEPRIO             8         8         4        31  Priority      Dynamic
ACP_SWAPFLGS            14        15         0        15  Boolean       Dynamic
```

Figure 18-2: Display Produced by SYSGEN Command SHOW/ACP

Figure 18-3 illustrates the hexadecimal display of the values of the ACP system parameters that the SHOW/ACP/HEX command produces.



```
SYSGEN>  SHOW/ACP/HEX

Parameters in use: Active
Parameter Name      Current    Default    Minimum    Maximum   Unit         Dynamic
--------------      -------    -------    -------    -------   ----         -------
ACP_MULTIPLE       00000001   00000001   00000000   00000001   Boolean      Dynamic
ACP_SHARE          00000001   00000001   00000000   00000001   Boolean
ACP_MAPCACHE       00000040   00000008   00000001   FFFFFFFF   Pages        Dynamic
ACP_HDRCACHE       000000B4   00000080   00000002   FFFFFFFF   Pages        Dynamic
ACP_DIRCACHE       0000008C   00000050   00000002   FFFFFFFF   Pages        Dynamic
ACP_WORKSET        00000000   00000000   00000000   FFFFFFFF   Pages        Dynamic
ACP_FIDCACHE       00000020   00000040   00000000   FFFFFFFF   File-Ids     Dynamic
ACP_EXTCACHE       00000040   00000040   00000000   FFFFFFFF   Extents      Dynamic
ACP_EXTLIMIT       000000C8   0000012C   00000000   000003E8   Percent/10   Dynamic
ACP_QUOCACHE       00000040   00000040   00000000   FFFFFFFF   Users        Dynamic
ACP_SYSACC         00000008   00000008   00000000   FFFFFFFF   Directories  Dynamic
ACP_MAXREAD        00000020   00000020   00000001   00000040   Blocks       Dynamic
ACP_WINDOW         00000007   00000007   00000001   FFFFFFFF   Pointers     Dynamic
ACP_WRITEBACK      00000001   00000001   00000000   00000001   Boolean      Dynamic
ACP_DATACHECK      00000003   00000002   00000000   00000003   Boolean      Dynamic
ACP_BASEPRIO       00000008   00000008   00000004   0000001F   Priority     Dynamic
ACP_SWAPFLGS       0000000E   0000000F   00000000   0000000F   Boolean      Dynamic
```

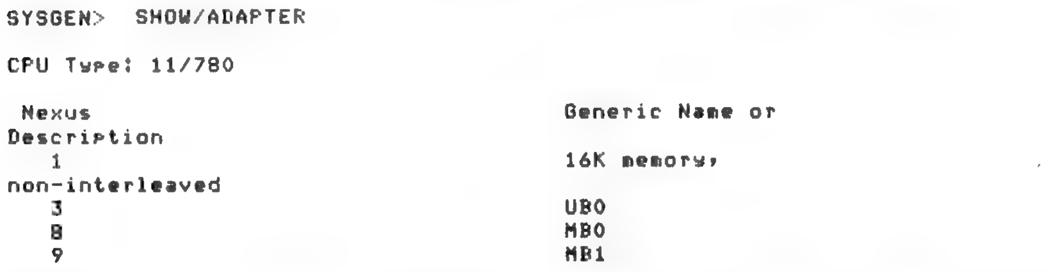
Figure 18-3: Display Produced by SYSGEN Command SHOW/ACP/HEX

18.2.20 SHOW (Adapter) Command

The SHOW (adapter) command lists all the nexus numbers and generic names on the adapter. It has the following format:

```
SHOW /ADAPTER
```

Figure 18-4 illustrates the display produced by SHOW/ADAPTER.



```
SYSGEN>  SHOW/ADAPTER

CPU Type: 11/780

 Nexus                                         Generic Name or
Description
   1                                           16K memory,
non-interleaved
   3                                           UB0
   8                                           MB0
   9                                           MB1
```

Figure 18-4: Display Produced by the SYSGEN Command SHOW/ADAPTER

Use of the SHOW (adapter) command requires the CMEXEC privilege.

18.2.21 SHOW (Configuration) Command

The SHOW (configuration) command displays information on the device configuration. It has the following format:

```
SHOW /CONFIGURATION
      [/ADAPTER=nexus]
      [/COMMAND_FILE]
      [/OUTPUT=file-spec]
```

nexus

Nexus number of the UNIBUS or MASSBUS adapter to be displayed. The nexus can be expressed as an integer or with one of the names listed by the SYSGEN command SHOW/ADAPTER.

file-spec

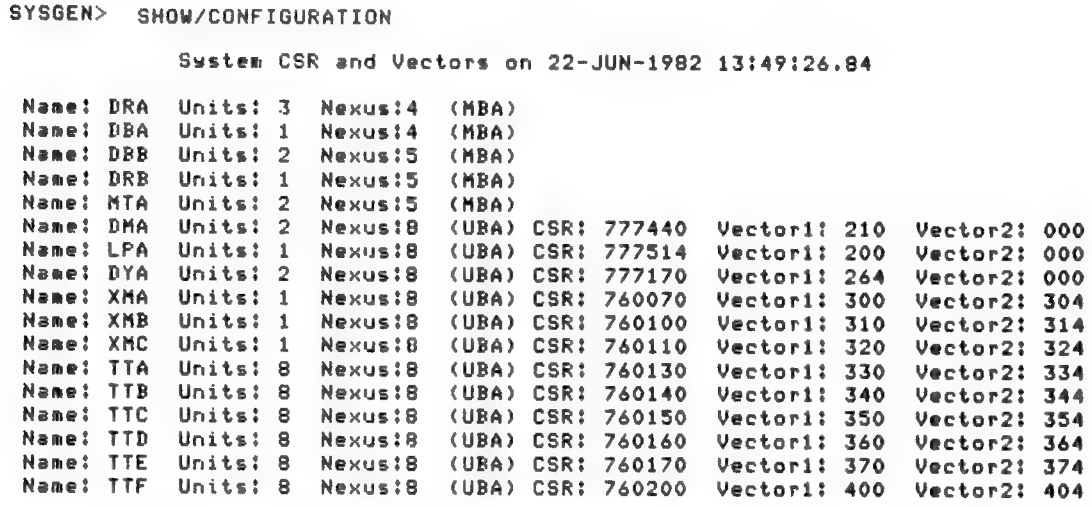
File specification on an optional output file. The default file type depends on the usage, as described below.

SHOW (configuration) shows devices by name, number of units, nexus number, and adapter type, as well as by CSR and vector addresses. You can specify an output file with the /OUTPUT qualifier. If you also specify the /COMMAND_FILE qualifier, SYSGEN formats all the device data into CONNECT (hardware) commands and writes the commands in an output file you specify. In this way, you can completely reconfigure a system without the use of the SYSGEN command AUTOCONFIGURE for UNIBUS devices.

Note that a device can be removed from the middle of the floating addresses without completely rejumpering the CSR and vector addresses of the remaining devices. For example, you would first modify your site-independent STARTUP.COM file to invoke the command file instead of issuing an AUTOCONFIGURE ALL command. Then issue the SHOW/CONFIGURATION/COMMAND_FILE/OUTPUT command to format and save the device data. Then, in the event you must bring the system down for service and remove a board, when the system reboots, SYS$SYSTEM:STARTUP.COM invokes your output file as a command procedure and the system automatically configures the system's UNIBUS devices and MASSBUS devices for you. Remember that a new version of SYS$SYSTEM:STARTUP.COM is provided with each major release, so any modifications you have made to SYS$SYSTEM:STARTUP.COM you would need to repeat after you install the new version. Although this technique can offer a convenient short-term solution, you should use AUTOCONFIGURE ALL whenever possible.

If you specify the /OUTPUT qualifier, but omit the file type, the default is LIS. However, if you specify /COMMAND_FILE and /OUTPUT qualifiers together, the default file type for the output file is COM.

Figure 18-5 illustrates the use of SHOW/CONFIGURATION to display the current system I/O data base on a VAX-11/780 processor:



```
SYSGEN>  SHOW/CONFIGURATION

             System CSR and Vectors on 22-JUN-1982 13:49:26.84

 Name: DRA  Units: 3  Nexus:4  (MBA)
 Name: DBA  Units: 1  Nexus:4  (MBA)
 Name: DBB  Units: 2  Nexus:5  (MBA)
 Name: DRB  Units: 1  Nexus:5  (MBA)
 Name: MTA  Units: 2  Nexus:5  (MBA)
 Name: DMA  Units: 2  Nexus:8  (UBA) CSR: 777440  Vector1: 210  Vector2: 000
 Name: LPA  Units: 1  Nexus:8  (UBA) CSR: 777514  Vector1: 200  Vector2: 000
 Name: DYA  Units: 2  Nexus:8  (UBA) CSR: 777170  Vector1: 264  Vector2: 000
 Name: XMA  Units: 1  Nexus:8  (UBA) CSR: 760070  Vector1: 300  Vector2: 304
 Name: XMB  Units: 1  Nexus:8  (UBA) CSR: 760100  Vector1: 310  Vector2: 314
 Name: XMC  Units: 1  Nexus:8  (UBA) CSR: 760110  Vector1: 320  Vector2: 324
 Name: TTA  Units: 8  Nexus:8  (UBA) CSR: 760130  Vector1: 330  Vector2: 334
 Name: TTB  Units: 8  Nexus:8  (UBA) CSR: 760140  Vector1: 340  Vector2: 344
 Name: TTC  Units: 8  Nexus:8  (UBA) CSR: 760150  Vector1: 350  Vector2: 354
 Name: TTD  Units: 8  Nexus:8  (UBA) CSR: 760160  Vector1: 360  Vector2: 364
 Name: TTE  Units: 8  Nexus:8  (UBA) CSR: 760170  Vector1: 370  Vector2: 374
 Name: TTF  Units: 8  Nexus:8  (UBA) CSR: 760200  Vector1: 400  Vector2: 404
```

Figure 18-5: Display Produced by SYSGEN Command SHOW/CONFIGURATION

Figure 18-6 illustrates a typical command file produced by a SHOW/CONFIGURATION/COMMAND_FILE/OUTPUT command. This command file could be used to automatically configure the system at a later date.

```
$ RUN SYS$SYSTEM:SYSGEN
AUTOCONFIGURE 4
AUTOCONFIGURE 5
CONNECT DMA0 /ADAP=8 /CSR=%0777440 /VECT=%0210 /NUMV=01 /DRIVER=DMDRIVER
CONNECT DMA1 /ADAP=8 /CSR=%0777440 /VECT=%0210 /NUMV=01 /DRIVER=DMDRIVER
CONNECT LPA0 /ADAP=8 /CSR=%0777514 /VECT=%0200 /NUMV=01 /DRIVER=LPDRIVER
CONNECT DYA0 /ADAP=8 /CSR=%0777170 /VECT=%0264 /NUMV=01 /DRIVER=DYDRIVER
CONNECT DYA1 /ADAP=8 /CSR=%0777170 /VECT=%0264 /NUMV=01 /DRIVER=DYDRIVER
CONNECT XMA0 /ADAP=8 /CSR=%0760070 /VECT=%0300 /NUMV=02 /DRIVER=XMDRIVER
CONNECT XMB0 /ADAP=8 /CSR=%0760100 /VECT=%0310 /NUMV=02 /DRIVER=XMDRIVER
CONNECT XMC0 /ADAP=8 /CSR=%0760110 /VECT=%0320 /NUMV=02 /DRIVER=XMDRIVER
CONNECT TTA0 /ADAP=8 /CSR=%0760130 /VECT=%0330 /NUMV=02 /DRIVER=DZDRIVER
CONNECT TTA1 /ADAP=8 /CSR=%0760130 /VECT=%0330 /NUMV=02 /DRIVER=DZDRIVER
CONNECT TTA2 /ADAP=8 /CSR=%0760130 /VECT=%0330 /NUMV=02 /DRIVER=DZDRIVER
       .
       .
       .
CONNECT TTF7 /ADAP=8 /CSR=%0760200 /VECT=%0400 /NUMV=02 /DRIVER=DZDRIVER
```

Figure 18-6: SHOW/CONFIGURATION/COMMAND_FILE/OUTPUT Command Example

Use of the SHOW (configuration) command requires the CMEXEC privilege.

18.2.22 SHOW (Device or Driver) Command

The SHOW (device or driver) command displays information on device drivers loaded into the system, the devices connected to them, and their I/O data bases. It has the following format:

```
SHOW    /DEVICE[=driver-name]
        /DRIVER[=driver-name]
```

driver-name
    Name of the driver; defaults to all device drivers loaded into the system.

The SHOW/DRIVER command displays the following information:

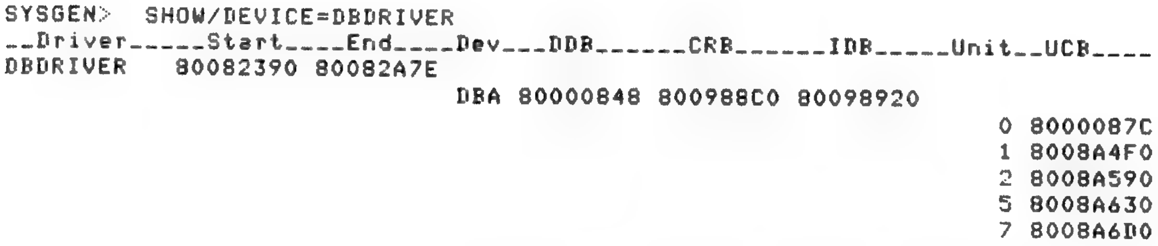
- Driver -- Name of the driver
- Start -- Starting address of the driver
- End -- Ending address of the driver

The SHOW/DEVICE command output provides the following additional information:

- Dev -- Name of each device connected to the driver
- DDB -- Address of the device's device data block
- CRB -- Address of the device's channel request block
- IDB -- Address of the device's interrupt dispatch block

- Unit -- Number of each unit on the device
- UCB -- Address of each unit's unit control block

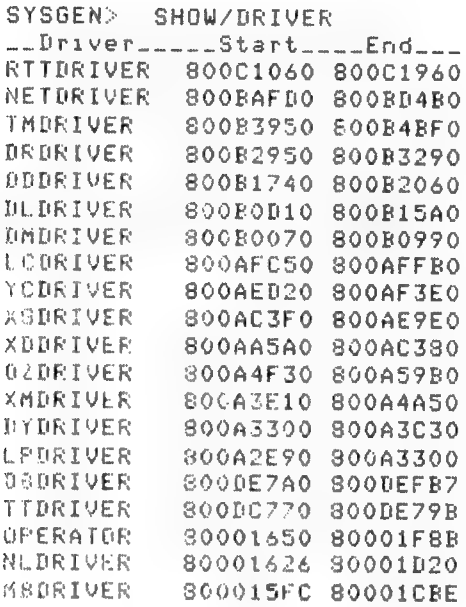
All addresses are in hexadecimal and are virtual. Figure 18-7 illustrates the SYSGEN command SHOW/DEVICE.



```
SYSGEN>  SHOW/DEVICE=DBDRIVER
__Driver_____Start____End____Dev___DDB______CRB______IDB_____Unit__UCB____
DBDRIVER   80082390 80082A7E
                             DBA 80000848 800988C0 80098920
                                                                0 8000087C
                                                                1 8008A4F0
                                                                2 8008A590
                                                                5 8008A630
                                                                7 8008A6D0
```

Figure 18-7: Display Produced by SYSGEN Command SHOW/DEVICE

Figure 18-8 illustrates a display of starting and ending addresses of all drivers that is produced by omitting the driver name on the SYSGEN command SHOW/DRIVER.



```
SYSGEN>  SHOW/DRIVER
__Driver_____Start____End___
RTTDRIVER  800C1060 800C1960
NETDRIVER  800BAFD0 800BD4B0
TMDRIVER   800B3950 800B4BF0
DRDRIVER   800B2950 800B3290
DDDRIVER   800B1740 800B2060
DLDRIVER   800B0D10 800B15A0
DMDRIVER   800B0070 800B0990
LCDRIVER   800AFC50 800AFFB0
YCDRIVER   800AED20 800AF3E0
XGDRIVER   800AC3F0 800AE9E0
XDDRIVER   800AA5A0 800AC380
DZDRIVER   800A4F30 800A59B0
XMDRIVER   800A3E10 800A4A50
DYDRIVER   800A3300 800A3C30
LPDRIVER   800A2E90 800A3300
DBDRIVER   800DE7A0 800DEFB7
TTDRIVER   800DC770 800DE79B
OPERATOR   80001650 80001F8B
NLDRIVER   80001626 80001D20
MBDRIVER   800015FC 80001CBE
```

Figure 18-8: Display Produced by SYSGEN Command SHOW/DRIVER

Use of the SHOW/DEVICE or SHOW/DRIVER command requires the CMEXEC privilege.

18.2.23 SHOW (Start-Up) Command

The SHOW (start-up) command displays the name of the current site-independent start-up command procedure. It has the following format:

```
SHOW /STARTUP
```

The following example illustrates the SYSGEN command SHOW (start-up):

```
     SYSGEN>  SHOW/STARTUP

      Startup command file = SYS$SYSTEM:STARTUP.COM
```

18.2.24 SHOW (UNIBUS) Command

The SHOW (UNIBUS) command displays the addresses in UNIBUS I/O space that can be addressed. It has the following format:

```
     SHOW/UNIBUS
            [/ADAPTER=nexus]
```

nexus
     Nexus number of the UNIBUS adapter whose address is to be displayed.

The nexus can be expressed as an integer or with one of the names listed by the SYSGEN command SHOW/ADAPTER.

If you do not specify a particular adapter, every UNIBUS is displayed.

NOTE

> The SHOW/UNIBUS command reads all device registers. For some controllers this may result in reading a character out of a buffer or produce some other undesired action. Thus, you should use the SHOW/UNIBUS command with caution, only when it is necessary to debug a UNIBUS configuration.

Figure 18-9 illustrates the use of the SHOW/UNIBUS command to display the available addresses for nexus 4. Note that use of the SHOW/UNIBUS command requires the CMKRNL privilege.

18.2.25 USE Command

The USE command initializes the SYSGEN work area with system parameter values and the name of the site-independent start-up command procedure. You specify the source for the both the parameter values and the procedure name. They can be retrieved from a parameter file, the current system image on disk, the active system in memory, or the default list. The USE command has the following format:

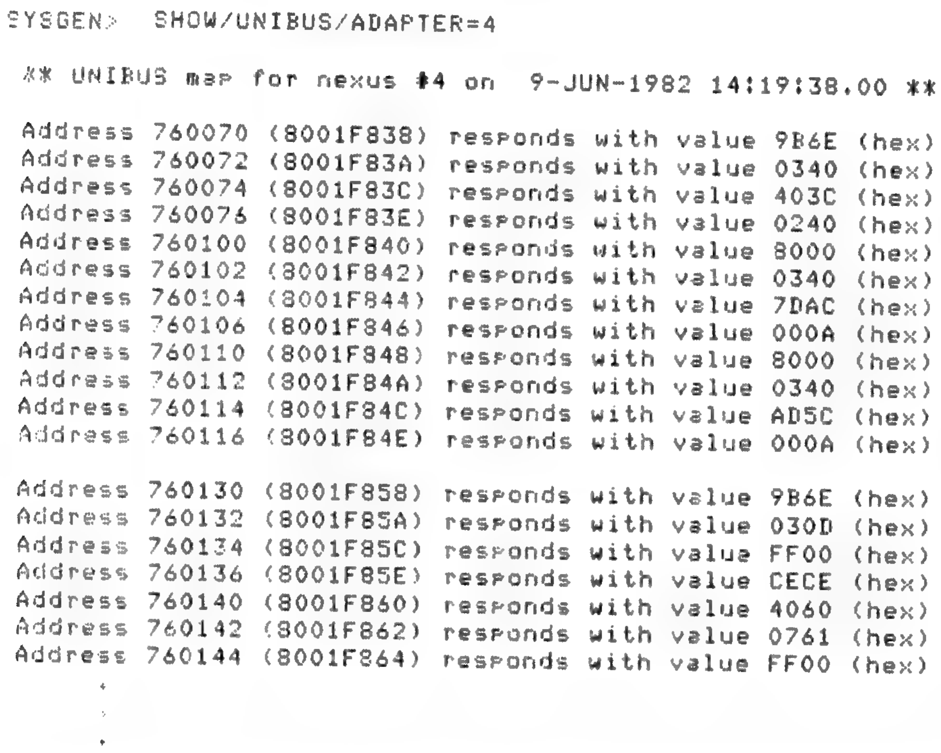
```
     USE  file-spec
          CURRENT
          ACTIVE
          DEFAULT
```

file-spec
     File specification of a system parameter file; the file type defaults to PAR.

You specify the file name of a parameter file or one of the keywords. The parameter file is either SYS$SYSTEM:AUTOGEN.PAR or the name of a parameter file you created with the SYSGEN command WRITE. Existing values in the SYSGEN work area are overwritten. The following command initializes the SYSGEN work area with parameter values that should allow VAX/VMS to boot on any standard configuration.

```
     SYSGEN>  USE DEFAULT
```

The initial values of the SYSGEN work area when the utility is invoked are the active values.



```
SYSGEN>  SHOW/UNIBUS/ADAPTER=4

 ** UNIBUS map for nexus #4 on  9-JUN-1982 14:19:38.00 **

 Address 760070 (8001F838) responds with value 9B6E (hex)
 Address 760072 (8001F83A) responds with value 0340 (hex)
 Address 760074 (8001F83C) responds with value 403C (hex)
 Address 760076 (8001F83E) responds with value 0240 (hex)
 Address 760100 (8001F840) responds with value 8000 (hex)
 Address 760102 (8001F842) responds with value 0340 (hex)
 Address 760104 (8001F844) responds with value 7DAC (hex)
 Address 760106 (8001F846) responds with value 000A (hex)
 Address 760110 (8001F848) responds with value 8000 (hex)
 Address 760112 (8001F84A) responds with value 0340 (hex)
 Address 760114 (8001F84C) responds with value AD5C (hex)
 Address 760116 (8001F84E) responds with value 000A (hex)

 Address 760130 (8001F858) responds with value 9B6E (hex)
 Address 760132 (8001F85A) responds with value 030D (hex)
 Address 760134 (8001F85C) responds with value FF00 (hex)
 Address 760136 (8001F85E) responds with value CECE (hex)
 Address 760140 (8001F860) responds with value 4060 (hex)
 Address 760142 (8001F862) responds with value 0761 (hex)
 Address 760144 (8001F864) responds with value FF00 (hex)
         .
         .
         .
```

Figure 18-9: Display Produced by SYSGEN Command SHOW/UNIBUS

18.2.26 WRITE Command

The WRITE command writes the system parameter values and the name of the site-independent start-up command procedure from the SYSGEN work area to your choice of a parameter file, the current system image on disk, or the active system in memory. (However, only the dynamic parameter values are written to the active system.)

The command has the following format:

```
     WRITE  file-spec
            CURRENT
            ACTIVE
```

file-spec
    File specification; the file type defaults to PAR.

Specification of a file results in the creation of a new parameter file, as illustrated in the following command:

```
     SYSGEN>  WRITE SYS$SYSTEM:SPECIAL
```

The next command modifies the current system image on disk (SYS$SYSTEM:SYS.EXE):

```
     SYSGEN>  WRITE CURRENT
```

Both the WRITE ACTIVE and WRITE CURRENT commands send a message to OPCOM and log the event. A message of the following form appears on the operator's console:

```
%OPCOM, 25-JUN-1982 16:04:06.30, message from user SYSTEM
%SYSGEN-I-WRITECUR, CURRENT system parameters modified by process ID
n into file y
```

Use of the WRITE ACTIVE command requires the CMKRNL privilege. Use of the WRITE CURRENT command requires the SYSPRV privilege.

18.3 ERROR MESSAGES

The VAX/VMS System Messages and Recovery Procedures Manual lists the messages generated by SYSGEN and provides explanations and suggested user actions.

CHAPTER 19

VERIFY UTILITY (ANALYZE/DISK_STRUCTURE)

The Verify Utility checks the readability and validity of Files-11 Structure Level 1 and Files-11 Structure Level 2 disk volumes and reports errors and inconistencies to the user. There are three operating modes: (1) error reporting with no repairs; (2) error reporting with repairs; (3) user-controlled selective repairs. Most classes of errors can be detected by default in one execution of the utility; the use of many qualifiers to select the error classes is not necessary.

19.1 RECOMMENDED USAGE

The Verify Utility is intended for use on a regular basis by system managers and operators to detect inconsistencies and errors in the file structure before additional damage to the files occurs. The identification of lost files (that is, files not cataloged in a directory) and those marked for deletion, and the deletion of unwanted files, can reclaim a significant amount of disk storage.

The most efficient way to use the utility is to execute it twice -- once without the /REPAIR qualifier to locate any possible errors and to evaluate these errors and the default error action, and again with the /REPAIR or /REPAIR/CONFIRM qualifiers to make the needed repairs.

If you find one or more errors when the utility is executed, you should refer to the list of error messages in Section 19.3 to determine utility and user actions.

To allow access to all portions of the file structure, the utility should always be executed with the user privilege BYPASS.

You should not execute the Verify Utility on a disk that has concurrent file activity. Use of the utility at this time will probably result in spurious messages that would incorrectly indicate the occurrence of severe file damage.

19.2 VERIFY UTILITY COMMAND STRING

The Verify Utility command string has the following format:

```
ANALYZE/DISK_STRUCTURE device-name:[/qualifier]
```

device-name:

Specifies the disk volume or volume set to be verified. A logical name may be used. If a volume set is specified, all volumes of the volume set must be mounted as Files-11 structured.

/qualifier(s)

The qualifiers that modify the ANALYZE/DISK_STRUCTURE operation. One or more qualifiers can be specified in each command string. Section 19.2.1 contains a complete description of each qualifier.

19.2.1 Command Qualifiers

Table 19-1 lists the Verify Utility command qualifiers.

Table 19-1: Verify Utility Command Qualifiers

Qualifier	Function
/CONFIRM /NOCONFIRM	If /CONFIRM is specified, the Verify Utility prompts you to confirm whether each repair should be performed. If the response is Y, the Verify Utility performs the repair. Otherwise, the repair is not performed. A few repairs allow you the option to delete the file. In these cases, if the response is D the Verify Utility deletes the file. If the response is Y, the Verify Utility performs the default repair. (For more information on messages, see Section 19.3.) The default is /NOCONFIRM.
/LIST[=file-spec] /NOLIST	If /LIST is specified, the Verify Utility produces a listing of the index file in file number order containing the file identification, file name, and owner UIC of each file. If file-spec is omitted, the default is SYS$OUTPUT. If you include a file-spec that does not have a file type, the command uses the default file type of LIS. No wild card characters are allowed in file-spec. The default is /NOLIST.
/READ_CHECK /NOREAD_CHECK	If /READ_CHECK is specified, the Verify Utility performs a read check of all allocated blocks on the file structure. The default is /NOREAD_CHECK.
/REPAIR /NOREPAIR	If /REPAIR is specified, the Verify Utility repairs errors that are detected in the file structure. The file structure is modified by the Verify Utility only if /REPAIR is specified. During execution with /REPAIR, the file structure is software write locked.

(continued on next page)

Table 19-1 (Cont.): Verify Utility Command Qualifiers

Qualifier	Function
	The default is /NOREPAIR, that is, the Verify Utility reports errors but does not repair them. NOTE If an error message is returned, the user should refer to the list of errors in Section 19.3 for utility and user actions.
/USAGE[=file-spec]	Specifies that a disk usage accounting file should be produced, in addition to the other specified functions of the Verify Utility. If all or part of the file specification is omitted, defaults are applied using the filespec SYS$DISK:USAGE.DAT. Section 19.2.1.1 describes the contents of the disk usage accounting file.

19.2.1.1 Disk Usage Accounting File - Disk usage accounting file records are of variable length. The first record of the file is an identification record (type USG$K_IDENT), followed by one file record (type USG$K_FILE) for each file on the volume. (USG$B_TYPE specifies the record type.) Table 19-2 lists the symbols defined by the $USGDEF macro.

Table 19-2: Disk Usage Accounting File Fields

Record Type	Field	Meaning
USG$K_IDENT	USG$L_SERIALNUM	Serial number of the volume. This value should be interpreted in octal radix.
	USG$T_STRUCNAME	12-byte volume set name (if the volume is part of a volume set). For a Files-11 Structure Level 1 volume, this field contains binary zeros; for a Files-11 Structure Level 2 volume that is not part of a volume set, this field contains spaces.
	USG$T_VOLNAME	12-byte volume name of relative volume 1.
	USG$T_OWNERNAME	12-byte volume owner name.

(continued on next page)

Table 19-2 (Cont.): Disk Usage Accounting File Fields

Record Type	Field	Meaning
	USG$T_FORMAT	12-byte volume format type. For a Files-11 Structure Level 1 volume, this field contains "DECFILE11A"; for a Files-11 Structure Level 2 volume, this field contains "DECFILE11B".
	USG$Q_TIME	Quadword system time when this usage file was created.
	USG$K_IDENT_LEN	USG$K_IDENT record length.
USG$K_FILE	USG$L_FILEOWNER	File owner UIC. USG$W_UICMEMBER is the member number; USG$W_UICGROUP is the group number.
	USG$L_ALLOCATED	Number of blocks allocated to the file, including file headers.
	USG$L_USED	Number of blocks used, up to and including the end of file block.
	USG$W_DIR_LEN	Length of the directory string portion of USG$T_FILESPEC, including the brackets.
	USG$W_SPEC_LEN	Length of the complete file specification in USG$T_FILESPEC.
	USG$T_FILESPEC	File specification, in the following format: [dir]nam.typ;ver This field is of variable length. A file that has more than one directory entry is listed under the first file specification found. A lost file has an empty directory string "[]" and the file name is taken from the file header. In some cases this information does not exist; application programs written to process the usage file must take this into consideration.
	USG$K_FILE_LEN	Maximum length of the FILE record.

19.3 VERIFY UTILITY MESSAGES

This section describes the information and error messages issued by the Verify Utility. Each message is followed by an explanation of its meaning and a recommendation for user action. A utility action, when given, is the action performed by the Verify Utility when /REPAIR has been specified. Unless otherwise indicated, /CONFIRM can be used to control whether the repair is to be made.

ADDQUOTA, error adding quota record for ['uic']

Explanation: The Verify Utility encountered an error during an attempt to add an entry to the quota file for the specified UIC. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

ALLOCCLR, blocks incorrectly marked allocated
LBN 'n' to 'n', RVN 'n'

Explanation: The Verify Utility found that the specified logical blocks on the specified relative volume were marked allocated in the storage bit map but were not allocated to a file.

Utility Action: The specified blocks are marked free in the storage bit map. The /CONFIRM qualifier is ignored in this action.

Severity: Information

ALLOCEXT, blocks allocated to lost extension file header
LBN 'n' to 'n', RVN 'n'

Explanation: The Verify Utility found that the specified logical blocks on the specified relative volume were allocated to a lost extension file header.

Utility Action: The specified blocks are marked free in the storage bit map. The /CONFIRM qualifier is ignored in this action.

Severity: Information

ALLOCMEM, error allocating virtual memory

Explanation: The Verify Utility encountered an error during an attempt to allocate dynamic virtual memory. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line. The account's page file quota or the SYSGEN parameter VIRTUALPAGECNT may need to be increased.

Severity: Fatal

ALLOCSET, blocks incorrectly marked free
LBN 'n' to 'n', RVN 'n'

Explanation: The Verify Utility found that the specified logical blocks on the specified relative volume were marked free in the storage bit map but were allocated to a file.

Utility Action: The specified blocks are marked allocated in the storage bit map. The /CONFIRM qualifier is ignored in this action.

Severity: Information

ALTIHDBAD, invalid alternate index file header, RVN 'n'

Explanation: The Verify Utility found that the alternate index file header on the specified volume was corrupted.

User Action: The alternate index file header is refreshed from the primary index file header.

Severity: Information

ASSIGN, error assigning channel to 'device-spec'

Explanation: The Verify Utility encountered an error during an attempt to assign a channel to the volume to be verified. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Fatal

BACKLINK, incorrect directory back link 'file-spec'

Explanation: The Verify Utility found that the back link of the specified file did not contain the file identification of the directory file in which the file is cataloged. This condition is normal if the file is cataloged in more than one directory.

User Action: If the file is cataloged in more than one directory, optionally use the /CONFIRM qualifier to direct the Verify Utility not to repair the back link.

Utility Action: Set the back link pointer to point to the directory file.

Severity: Information

BADBITMAP, BITMAP.SYS too small or not contiguous, RVN 'n'

Explanation: The Verify Utility found that the storage bit map file, [000000]BITMAP.SYS;1, on the specified relative volume, was not large enough to accommodate the volume size and volume cluster factor, or that it was not contiguous. This error can result from an erroneous volume size in the storage control block or an erroneous volume cluster factor in the home block.

User Action: If the volume can no longer be mounted by the operating system, it is corrupted and must be reconstructed from back-up media. If the volume can be mounted, the damage can be repaired by copying the volume to fresh media.

Severity: Fatal

BADDIR, directory ['directory'] has invalid format

Explanation: The Verify Utility found that the specified directory file is not in the expected format for a directory file. The file is corrupted.

Utility Action: Processing of the directory file is terminated. If the Verify Utility finds lost files, it neither repairs nor reports them until the error is corrected.

User Action: Delete the directory file and reenter the Verify Utility command line.

Severity: Information

BADDIRENT, invalid file identification in directory entry
'file-spec'

Explanation: The Verify Utility found that the specified directory entry did not contain a valid file identification, meaning that the file has probably been deleted.

Utility Action: The directory entry is removed.

Severity: Information

BADEFBLK, file ('file-id') 'file-name'
inconsistent EFBLK and map area

Explanation: The Verify Utility found that the end of file pointer recorded in the VAX-11 RMS record attributes for the specified file did not lie within the space actually allocated to the file.

Utility Action: The end of file pointer is set to point to the end of the allocated space.

Severity: Information

BADHEADER, file ('file-id') 'file-name'
invalid file header

Explanation: The Verify Utility found that the specified file header was corrupted.

Utility Action: The file header is rewritten with a deleted file header.

User Action: If the specified file header is currently in use, the file can no longer be accessed by the operating system and must be reconstructed from back-up media. If the file header is not in use, no action is necessary.

Severity: Information

BADHIBLK, file ('file-id') 'file-name'
inconsistent HIBLK and map area

Explanation: The Verify Utility found that the file size recorded in the VAX-11 RMS record attributes did not agree with the number of blocks actually allocated to the file.

Utility Action: The record attributes are corrected by changing the file size to reflect the correct number of blocks.

Severity: Information

BBLHEADER, file ('file-id') 'file-name'
contains suspected bad blocks

Explanation: The Verify Utility found that the specified file was marked as containing suspected bad blocks. When the file is deleted, the blocks will be verified and returned to the bad block file if necessary.

Utility Action: None.

User Action: If desired, examine the contents of the file and reconstruct from back-up media as necessary.

Severity: Information

CHKALTHOME, invalid alternate home block, VBN 'n', RVN 'n'

Explanation: The Verify Utility found that the alternate home block at the specified virtual block of the index file on the specified relative volume was corrupted.

Utility Action: The alternate home block is refreshed from the primary home block.

Severity: Information

CHKPRIHOME, invalid primary home block, VBN 'n', RVN 'n'

Explanation: The Verify Utility found that the primary home block at the specified virtual block of the index file on the specified relative volume was corrupted.

Utility Action: None.

User Action: If the volume can no longer be mounted by the operating system, it is corrupted and must be reconstructed from backup media. If the volume can be mounted, the damage can be repaired by copying the volume to fresh media.

Severity: Information

CHKSCB, invalid storage control block, RVN 'n'

Explanation: The Verify Utility found that the storage control block on the specified relative volume was corrupted.

Utility Action: The block is reconstructed.

User Action: If the volume can no longer be mounted by the operating system, it is corrupted and must be reconstructed from back-up media. If the volume can be mounted, the damage can be repaired by copying the volume to fresh media.

Severity: Information

CREATELOST, error creating directory [SYSLOST]

Explanation: The Verify Utility encountered an error during an attempt to create the directory [SYSLOST] to enter lost files. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

DELETE, file ('file-id') 'file-name'
error deleting file

Explanation: The Verify Utility encountered an error during an attempt to delete the specified file. The accompanying messagge provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

DELHEADER, file ('file-id') 'file-name'
marked for delete

Explanation: The Verify Utility found that the specified file was marked for deletion, but was not deleted.

Utility Action: The file is deleted.

Severity: Information

DIRNAME, directory file 'file-spec' is not named '.DIR;1'

Explanation: The Verify Utility found that the specified file was marked with the directory file characteristic, but did not have a file type of DIR and a version number of 1. VAX-11 RMS cannot recognize the file as a directory.

User Action: If necessary, use the RENAME command to rename the file.

Severity: Information

DSAQUOTA, error disabling quota processing

Explanation: The Verify Utility encountered an error during an attempt to disable quota processing. The accompanying message provides additional information.

Utility Action: None. Quota processing remains enabled.

User Action: Correct the condition that caused the error and use the Disk Quota Utility to disable quota processing.

Severity: Information

ENAQUOTA, error enabling quota processing

Explanation: The Verify Utility encountered an error during an attempt to enable quota processing. The accompanying message provides additional information.

Utility Action: The Verify Utility will not reconstruct disk quota information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

ENTERLOST, file ('file-id') 'file-name'
error entering file in directory [SYSLOST]

Explanation: The Verify Utility encountered an error during an attempt to enter the specified file in directory [SYSLOST]. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

FINDHOME, no valid home block, RVN 'n'

Explanation: The Verify Utility could not locate a valid home block on the specified relative volume.

User Action: If the volume can no longer be mounted by the operating system, it is corrupted and must be reconstructed from back-up media. If the volume can be mounted, this message indicates a software error in the Verify Utility. In this case, submit a Software Performance Report (SPR) to DIGITAL.

Severity: Fatal

FINDIHD, no valid index file header, RVN 'n'

Explanation: The Verify Utility found that both the primary and alternate index file headers were corrupted.

User Action: If the volume can no longer be mounted by the operating system, it is corrupted and must be reconstructed from back-up media. If the volume can be mounted, this message indicates a software error in the Verify Utility. In this case, submit an SPR.

Severity: Information

FREEMEM, error freeing virtual memory

Explanation: The Verify Utility encountered an error during an attempt to free dynamic virtual memory. The accompanying message provides additional information. This message indicates a software error in the Verify Utility.

User Action: Submit an SPR.

Severity: Fatal

FUTBAKDAT, file ('file-id') 'file-name'
    backup date is in the future

Explanation: The Verify Utility found that the date of the last back-up recorded for the file was later than the current date.

Utility Action: The back-up date is set to the current date.

Severity: Information

FUTCREDAT, file ('file-id') 'file-name'
    creation date is in the future

Explanation: The Verify Utility found that the creation date recorded for the file was later than the current date.

Utility Action: The creation date is set to the current date.

Severity: Information

FUTREVDAT, file ('file-id') 'file-name'
revision date is in the future

Explanation: The Verify Utility found that the date of the last modification recorded for the file was later than the current date.

Utility Action: The revision date is set to the current date.

Severity: Information

GETDVI, error getting device characteristics, RVN 'n'

Explanation: The Verify Utility encountered an error in an attempt to get the device characteristics of the specified relative volume. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Fatal

INCQUOTA, QUOTA.SYS indicates 'n' blocks used, actual use is 'n'
blocks for ['uic']

Explanation: The Verify Utility found that the disk usage for the specified UIC as recorded in the quota file did not agree with the actual disk usage.

Utility Action: The quota file is corrected.

Severity: Information

INVDEVICE, invalid device 'device-spec'

Explanation: An invalid device was specified in the command.

User Action: Reenter the command specifying a valid device. The specification must reference a disk device and must not contain any file specification components other than a device name.

Severity: Fatal

INVEXTBACK, file ('file-id') 'file-name'
invalid extension file header back link

Explanation: The Verify Utility has found that the back link in an extension file header for the specified file does not point to the primary file header.

Utility Action: The back link is changed to point to the primary file header.

Severity: Information

INVEXTFID, file ('file-id') 'file-name'
invalid extension file header forward link

Explanation: The Verify Utility has found that the forward link in a file header to the next extension file header does not point to a valid file header.

Utility Action: The link is cleared. Part of the file cannot be accessed by the operating system.

User Action: Examine and delete the file. Reconstruct the file from back-up media.

Severity: Information

INVEXTHDR, file ('file-id') 'file-name'
invalid extension file header sequence

Explanation: The Verify Utility has found that the sequence numbers in the extension file headers for the specified file are not correct. The sequence number should increase by 1 in each extension file header.

Utility Action: The link that points to the erroneous extension file header is cleared. Part of the file cannot be accessed by the operating system.

User Action: Examine and delete the file. Reconstruct the file from back-up media.

Severity: Information

LOCKHEADER, file ('file-id') 'file-name'
deaccess locked

Explanation: The Verify Utility found that the specified file was marked as deaccess locked.

User Action: If the /CONFIRM qualifier was specified, the user may elect to unlock or to delete the file.

Utility Action: The file is unlocked. Optionally, if the user responded to the /CONFIRM qualifier prompt by typing D, the file is deleted.

Severity: Information

LOCKVOL, error locking volume set - /REPAIR cancelled

Explanation: The Verify Utility encountered an error during an attempt to lock the volume set to execute a /REPAIR operation. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line. The /REPAIR qualifier requires a system UIC or SYSPRV privilege, or that the user is the owner of the volume.

Severity: Information

LOSTEXTHDR, file ('file-id') 'file-name'
lost extension file header

Explanation: The Verify Utility found that an extension file header is not in the extension file header chain of any file.

Utility Action: The file header is rewritten with a deleted header and the blocks allocated to the file header are marked free. The /CONFIRM qualifier is ignored in this action.

Severity: Information

LOSTHEADER, file ('file-id') 'file-name'
not found in a directory

Explanation: The Verify Utility found that the specified file was not cataloged in a directory. This error is neither reported nor repaired if any errors occurred in processing the directories on the volume set.

User Action: If the /CONFIRM qualifier is specified, the user may elect to catalog the file in directory [SYSLOST] or to delete the file.

The user may want to rerun the Verify Utility after repairs have been made.

Utility Action: The file is cataloged in directory [SYSLOST]. Optionally, if the user responded to the /CONFIRM qualifier prompt by typing D, the file is deleted.

Severity: Information

LOSTSCAN, due to directory errors, lost files will not be entered

Explanation: The Verify Utility encountered errors during the directory scan that were reported by previous messages. Therefore, lost files will not be entered in directory [SYSLOST].

User Action: Refer to the suggested user actions for the previously reported errors, correct the errors, and reenter the Verify Utility command line.

Severity: Information

MAPAREA, file ('file-id') 'file-name'
invalid map area

Explanation: The Verify Utility found that the specified file has a corrupted map area.

Utility Action: None.

User Action: The damage can probably be repaired by copying the file to a new version and deleting the original version. It may be necessary to reconstruct the file from back-up media.

Severity: Information

MAXVOLS, too many volumes in volume set

Explanation: The Verify Utility cannot process a volume set that contains more than 255 volumes. The volume set count in the home block on relative volume 1 is probably corrupted.

User Action: Reconstruct the volume set from back-up media.

Severity: Fatal

MODQUOTA, error modifying quota record for ['uic']

Explanation: The Verify Utility encountered an error during an attempt to modify the usage data in the quota file entry for the specified UIC. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

```
MULTALLOC, file ('file-id') 'file-name'
        multiply allocated blocks
        VBN 'n' to 'n'
        LBN 'n' to 'n', RVN 'n'
```

Explanation: The Verify Utility found that the specified logical blocks on the specified relative volume were allocated to more than one file. These blocks are the specified virtual blocks of the specified file.

Utility Action: None.

User Action: Examine all the multiply allocated block messages to determine the files involved in each instance of multiple allocation. Then, without allowing other file activity, perform the following functions:

- Copy all but one file involved in each instance to a new version.
- Delete the version of each file that was copied that contains the multiple allocation. This action marks blocks free in the storage bit map that are not in fact free blocks.
- Reenter the Verify Utility command line with the /REPAIR qualifier to correct the storage bitmap.
- Examine each file for corruption and reconstruct from backup media as necessary.

Severity: Information

MULTEXTHDR, file ('file-id') 'file-name'
multiply allocated extension file header

Explanation: The Verify Utility found that the specified extension file header was in the extension header chain of more than one file.

Utility Action: None.

User Action: The damage can be repaired by copying the volume to fresh media or by copying all of the files to new versions and then deleting the original versions. Examine each file for corruption and reconstruct from back-up media as necessary.

Severity: Information

NOREPAIR, one or more volumes write locked - /REPAIR cancelled

Explanation: The Verify Utility found that one or more volumes of the volume set are write locked, and therefore a /REPAIR operation cannot be executed.

User Action: Write enable all volumes of the volume set and reenter the Verify Utility command line.

Severity: Information

OPENBITMAP, error opening BITMAP.SYS, RVN 'n'

Explanation: The Verify Utility encountered an error during an attempt to open the storage bit-map file, [000000]BITMAP.SYS;1, on the specified relative volume. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Fatal

OPENDIR, error opening directory ['directory']

Explanation: The Verify Utility encountered an error during an attempt to open the specified directory. The accompanying message provides additional information.

Utility Action: If the Verify Utility finds lost files, it neither repairs nor reports them until the error is corrected.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

OPENFILE, file (file-id') 'file-name'
error opening file for read check

Explanation: The Verify Utility encountered an error during an attempt to open the specified file to execute the /READ_CHECK function. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

OPENINDEX, error opening INDEXF.SYS, RVN 'n'

Explanation: The Verify Utility encountered an error during an attempt to open the index file, [000000]INDEXF.SYS;1, on the specified relative volume. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Fatal

OPENQUOTA, error opening QUOTA.SYS

Explanation: The Verify Utility encountered an error during an attempt to open the quota file, [000000]QUOTA.SYS;1, on relative volume 1. The accompanying message provides additional information. This message is normal if quotas are not enforced on the volume.

User Action: If necessary correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

PRIIHDBAD, invalid primary index file header, RVN 'n'

Explanation: The Verify Utility found that the primary index file header on the specified relative volume was corrupted.

Utility Action: The primary index file header is refreshed from the alternate index file header.

Severity: Information

READBOOT, error reading boot block, RVN 'n'

Explanation: The Verify Utility encountered an error during an attempt to read the boot block from virtual block 1 of the index file, [000000]INDEXF.SYS;1, on the specified relative volume. The accompanying message provides additional information.

User Action: Note that the volume probably cannot be used as a system volume for a VAX-11 processor model that requires access to the boot block during a bootstrap operation.

Severity: Information

READDIR, error reading directory ['directory']

Explanation: The Verify Utility encountered an error during an attempt to read the specified directory. The accompanying message provides additional information.

Utility Action: If the Verify Utility finds lost files, it neither repairs nor reports them, until the error is corrected.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

READFILE, file ('file-id') 'file-name'
error reading VBN 'n'

Explanation: The Verify Utility encountered an error during an attempt to read the specified virtual block of the specified file while executing the /READ_CHECK function. The accompanying message provides additional information.

User Action: Reconstruct the file from back-up media if necessary.

Severity: Information

READHEADER, file ('file-id') 'file-name'
file header read error

Explanation: The Verify Utility encountered an error during an attempt to read the specified file header from the index file, [000000]INDEXF.SYS;1, on the specified relative volume. The accompanying message provides additional information.

Utility Action: The file header is rewritten with a deleted file header. The /CONFIRM qualifier is ignored in this action.

User Action: If the specified file header is currently in use, the file can no longer be accessed by the operating system and must be reconstructed from back-up media. If the file header is not in use, no action is necessary because the operating system will not attempt to use the file header.

Severity: Information

READHOME, error reading home block, VBN 'n', RVN 'n'

Explanation: The Verify Utility encountered an error during an attempt to read the home block from the specified virtual block of the index file, [000000]INDEXF.SYS;1, on the specified relative volume. The accompanying message provides additional information.

Utility Action: The Verify Utility reconstructs the block.

User Action: The volume probably can be mounted by the operating system using one of the alternate home blocks. If the error persists, the user should consider copying the volume to fresh media. If /REPAIR is specified, the Verify Utility rewrites the specified virtual block. This may clear the read error. If the error cannot be cleared, the user must then rewrite the volume.

Severity: Information

READIBMAP, error reading index file bitmap, VBN 'n', RVN 'n'

Explanation: The Verify Utility encountered an error during an attempt to read the index file bit map from the specified virtual block of the index file, [000000]INDEXF.SYS;1, on the specified relative volume. The accompanying message provides additional information.

Utility Action: The Verify Utility accepts the data as read.

User Action: The volume probably can be mounted by the operating system. However, if the error persists, the user should copy the volume to fresh media or reconstruct the volume from back-up media. If /REPAIR is specified, the Verify Utility rewrites the specified virtual block. This may clear the read error. If the error cannot be cleared, the user must then rewrite the volume.

Severity: Information

READQUOTA, error reading QUOTA.SYS, VBN 'n'

Explanation: The Verify Utility encountered an error during an attempt to read the specified virtual block of the quota file, [000000]QUOTA.SYS;1, on relative volume 1. The accompanying message provides additional information.

Utility Action: The Verify Utility disregards quota information in the block.

User Action: If necessary, correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

READSBMAP, error reading storage bitmap, VBN 'n', RVN 'n'

Explanation: The Verify Utility encountered an error during an attempt to read the storage bitmap from the specified virtual block of the storage bitmap file, [000000]BITMAP.SYS;1, on the specified relative volume. The accompanying message provides additional information.

User Action: The volume probably can be mounted by the operating system. However, if the error persists, the user should copy the volume to fresh media or reconstruct the volume from back-up media. If /REPAIR is specified, the Verify Utility rewrites the specified virtual block. This may clear the read error. If the error cannot be cleared, the user must then rewrite the volume.

Severity: Information

READSCB, error reading storage control block, RVN 'n'

Explanation: The Verify Utility encountered an error during an attempt to read the storage control block from virtual block 1 of the storage bit map file, [000000]BITMAP.SYS;1, on the specified relative volume. The accompanying message provides additional information.

User Action: The volume probably can be mounted by the operating system. However, if the error persists, the user should copy the volume to fresh media or reconstruct the volume from back-up media. If /REPAIR is specified, the Verify Utility rewrites the specified virtual block. This may clear the error. If the error cannot be cleared, the user must then rewrite the volume.

Severity: Information

REMOVE, file ('file-id')
    error removing directory entry

Explanation: The Verify Utility encountered an error during an attempt to remove a directory entry referencing the specified file identification. The accompanying message provides additional information.

User Action: Correct the condition that caused the error and reenter the Verify Utility command line.

Severity: Information

UNLKVOL, error unlocking volume set

Explanation: The Verify Utility encountered an error during an attempt to unlock the volume set to allow allocation. The accompanying message provides additional information.

Utility Action: None. The volume set remains locked.

User Action: Correct the condition that caused the error and then dismount and remount the volume set.

Severity: Information

WRITEHEADER, file ('file-id') 'file-name'
file header write error

Explanation: The Verify Utility encountered an error during an attempt to write the specified file header to the index file, [000000]INDEXF.SYS;1, on the specified relative volume. The accompanying message provides additional information.

User Action: If the specified file header is currently in use, the file can no longer be accessed by the operating system and must be deleted and reconstructed from back-up media. If the file header is not in use, no action is necessary because the operating system will not attempt to use the file header.

Severity: Information

WRITEHOME, error writing home block, VBN 'n' RVN 'n'

Explanation: The Verify Utility encountered an error during an attempt to write the home block to the specified virtual block of the index file, [000000]INDEXF.SYS;1, on the specified relative volume. The accompanying message provides additional information.

User Action: The volume probably can be mounted by the operating system using one of the alternate home blocks, but the user should consider copying the volume to fresh media.

Severity Information

WRITEIBMAP, error writing index file bitmap, VBN 'n', RVN 'n'

Explanation: The Verify Utility encountered an error during an attempt to write the index file bitmap to the specified virtual block of the index file, [000000]INDEXF.SYS;1, on the specified relative volume. The accompanying message provides additional information.

User Action: The volume probably can be mounted by the operating system, but the user should copy the volume to fresh media or reconstruct the volume from back-up media.

Severity: Information

WRITESBMAP, error writing storage bitmap, VBN 'n', RVN 'n'

Explanation: The Verify Utility encountered an error during an attempt to write the storage bitmap to the specified virtual block of the storage bitmap file, [000000]BITMAP.SYS;1, on the specified relative volume. The accompanying message provides additional information.

User Action: The volume probably can be mounted by the operating system, but the user should copy the volume to fresh media or reconstruct the volume from back-up media.

Severity: Information

WRITESCB, error writing storage control block, RVN 'n'

Explanation: The Verify Utility encountered an error during an attempt to write the storage control block to virtual block 1 of the storage bit map file, [000000]BITMAP.SYS;1, on the specified relative volume. The accompanying message provides additional information.

User Action: The volume probably can be mounted by the operating system, but the user should copy the volume to fresh media or reconstruct the volume from back-up media.

Severity: Information

WRONGOWNER, file ('file-id') 'file-name'
inconsistent extension file header owner UIC

Explanation: The Verify Utility found that the file owner UIC recorded in an extension file header for the specified file did not agree with that recorded in the primary header.

Utility Action: The extension file header is corrected.

Severity: Information

APPENDIX A

FILES-11 DEVICES SUPPORTED BY VAX/VMS

Tables A-1 and A-2 list the Files-11 structured devices supported by VAX/VMS, with their storage characteristics and the device codes used to refer to them. Table A-1 lists magnetic tape devices; Table A-2 lists disk devices. See the *VAX/VMS I/O User's Guide* for more information on these devices.

Table A-1: Magnetic Tape Devices

Controller	Drive	Code	No. of Tracks	Recording Density (bpi)	Tape Speed (ips)	Max. Data Transfer Rate in Bytes Per Second	Recording Method[1]
TS11	TS04	MS	9	1600	45	72,000	PE
TM03	TE16	MT	9	800 or 1600	45	36,000 (for 800 bpi); 72,000 (for 1600 bpi)	NRZI or PE
	TU45	MT	9	800 or 1600	75	60,000 (for 800 bpi); 120,000 (for 1600 bpi)	NRZI or PE
	TU77	MT	9	800 or 1600	125	100,000 (for 800 bpi); 200,000 (for 1600 bpi)	NRZI or PE
TM78	TU78	MF	9	1600 or 6250	125	200,000 (for 1600 bpi); 781,250 (for 6250 bpi)	PE or GCR

[1] NRZI = non-return-to-zero-inverted; PE = phase encoded; GCR = group coded recording

Table A-2: Disk Devices

Model	Code	Type[1]	RPM	Surfaces	Cylinders	Bytes/ Track	Bytes/ Drive
RB02	DQ	Cart	2400	2	512	10,240	10,485,760
RL02	DL	Cart	2400	2	512	10,240	10,485,760
RM03	DR	Pack	3600	5	823	16,384	67,420,160
RM05	DR	Pack	3600	19	823	16,384	256,196,608
RB80	DQ	Fix	3600	14	559	15,872	124,214,272
RM80	DR	Fix	3600	14	559	15,872	124,214,272
RP05	DB	Pack	3600	19	411	11,264	87,960,576
RP06	DB	Pack	3600	19	815	11,264	174,423,040
RP07	DR	Fix	3600	16 [5]	630	25,600	516,096,000
RK06	DM	Cart	2400	3	411	11,264	13,888,512
RK07	DM	Cart	2400	3	815	11,264	27,540,480
RX01	DX	Flop	360	1	77	3,328	256,256
RX02	DY	Flop	360	1	77	3,328[2] 6,656[3]	256,256[2] 512,512[3]
TU58#	DD	Cart	--[4]	--[4]	--[4]	--[4]	262,144

1. Pack = pack disk; Cart = cartridge disk; Flop = floppy (flexible diskette); FIX = fixed media

2. Single density

3. Double density

4. A magnetic tape device, the TU58 operationally resembles a disk device

5. The RP07 has 16 surfaces but 32 tracks per cylinder

APPENDIX B

BACKUP MEDIA FORMATS

BACKUP tapes are ANSI labeled, according to ANSI standard X3.27-1977. Each save set you create corresponds to a file on the tape. The save-set name and type are mapped into the name field in the tape header label. This operation resembles the manner in which file names are mapped into the tape file header label by the VAX/VMS magnetic tape ancillary control process (ACP).

The save-set file is formatted in fixed-length records, with each record occupying one tape block.

The owner identifier field of volume 1 is written with the characters D%C, followed by the tape's owner UIC and protection as specified by the /OWNER_UIC and /PROTECTION qualifiers. For more information on labels see the *VAX-11 Record Management Services Reference Manual*.

B.1 SAVE-SET FILE FORMAT

The blocks within a save-set file are numbered in sequence. This allows BACKUP to detect tape mispositioning and lost blocks; it also allows unrecoverable write errors to be retried. Rewritten data blocks are detected while the tape is being read by noting duplicate sequence numbers. An exclusive-OR (XOR) block is written at the end of an XOR redundancy group. The purpose of the XOR block is to reconstruct data lost in a block. In each XOR redundancy group, an XOR operation is performed on all data blocks to form the XOR block. If one of the blocks cannot be restored from the back-up medium, BACKUP takes the XOR of the remaining good blocks. When an XOR operation is performed on the XOR of good blocks and the XOR block, the data in the missing block is reproduced. Recovery of bad data will not work if more than one block in an XOR redundancy group is corrupted. The XOR block is the last block in the XOR redundancy group.

The save-set file starts with a BACKUP summary record that contains general summary information about the back-up. Much of the save-set is devoted to file data. Each file is represented by a file attribute record that describes the file; this is followed by file data records containing the blocks of the file. The organization of a save-set can be displayed by using the command format:

```
     BACKUP/LIST/ANALYZE save-set-specifier
```

B.2 BLOCK AND RECORD FORMAT SUMMARY

Each tape block consists of a block header and one or more data records.

The block header is 256 bytes long. The header contains information used to identify the tape block and to implement various reliability features. The information in the block header allows the file data contained in the tape block to be retrieved and identified independently of all other blocks on the tape.

Attribute records or data records are stored immediately after the block header. All records are preceded by a 16-byte record header that identifies the record and specifies its length. Attribute records contain summary data or information used to rebuild files and physical volumes. Data records contain the actual data of the files being saved. The size of the data record is a multiple of 512 bytes (plus 16 bytes for the record header).

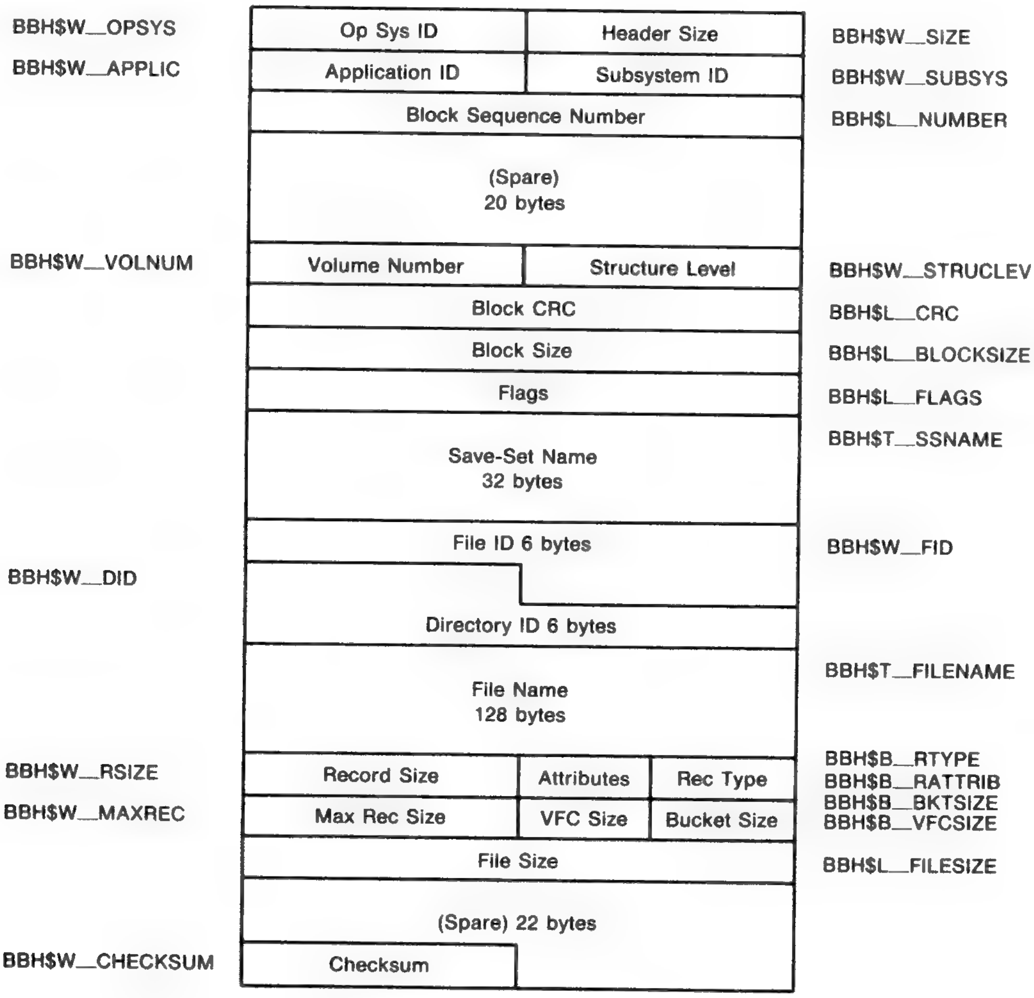
The length of each block on the tape can be computed by the formula:

```
length = 256 + 16 + (n * 512)
```

n is the total number of 512-byte data units in the block.

B.3 BLOCK HEADER

A description of the block header formats is shown in Figure B-1 and in Table B-1.



ZK-869-82

Figure B-1: Block Header

Table B-1: Contents of the Block Header

Field Name	Contents
BBH$W_SIZE	Contains the size of the block header area (in bytes). BACKUP writes the value 256 in this field.
BBH$W_OPSYS	Contains the identification number of the operating system writing the tape. BACKUP writes the value 1024 in this field.
BBH$W_SUBSYS	Contains the identification number of the subsystem writing the tape. BACKUP writes the value 1 in this field.
BBH$W_APPLIC	Contains the identification number of the application writing the tape. BACKUP writes the value 1 in this field for normal data blocks, or the value 2 for XOR blocks.
BBH$L_NUMBER	Contains the number of the save-set block. The save-set blocks are numbered sequentially, starting with 1.
BBH$W_STRUCLEV	Contains the structure level of the version of BACKUP that wrote the block header. For level 1, version 1, the value is 257.
BBH$W_VOLNUM	Contains the volume number of the BACKUP medium. BACKUP tape volumes (reels) are numbered starting with 1.
BBH$L_CRC	Contains the CRC computed on the entire data block using the AUTODIN II CRC algorithm. The CRC field is set to 0 during the computation. If the flag bit BBH$V_NOCRC is set, this field contains 0.
BBH$L_BLOCKSIZE	Contains the size of the block in bytes.
BBH$L_FLAGS	Contains the following flag bits for the block: BBH$V_NOCRC Set if no CRC was computed for this block.
BBH$T_SSNAME	Contains the file name of the save set, stored as a counted ASCII string.
BBH$W_FID	Contains the file identification of the file currently being copied to the BACKUP tape.
BBH$W_DID	Contains the directory identification of the file currently being copied to the BACKUP tape.
BBH$T_FILENAME	Contains the name of the file currently being copied to the BACKUP tape. The name appears as a counted ASCII string, and includes the complete directory string of the file, but not the device name.

(Continued on next page)

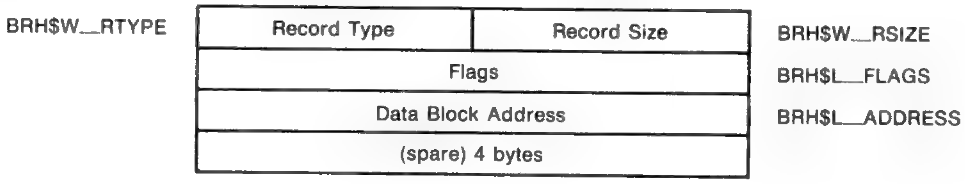
Table B-1 (Cont.): Contents of the Block Header

Field Name	Contents
BBH$B_RTYPE	Contains the record type of the current file.[1]
BBH$B_RATTRIB	Contains the record attributes of the current file.[1]
BBH$W_RSIZE	Contains the record size of the current file.[1]
BBH$B_BKTSIZE	Contains the bucket size of the current file.[1]
BBH$B_VFCSIZE	Contains the VFC area size for the current file.[1]
BBH$W_MAXREC	Contains the maximum record size of the current file.[1]
BBH$L_FILESIZE	Contains the allocated size of the current file.
BBH$W_CHECKSUM	Contains a checksum of the block header computed by the CRC-16 algorithm.

[1] For more information on these fields, see the VAX/VMS I/O User's Guide.

B.4 RECORD HEADER

A description of the record header format is shown in Figure B-2 and in Table B-2.



ZK-870-82

Figure B-2: Record Header

Table B-2: Contents of the Record Header

Field Name	Contents
BRH$W_RSIZE	Contains the size of the data portion of the record, in bytes.
BRH$W_RTYPE	Contains the type code of this record. The type codes are: BRH$K_NULL — Indicates a null record. Used as filler when necessary.

(Continued on next page)

Table B-2 (Cont.): Contents of the Record Header

Field Name	Contents	
BRH$W_RTYPE (Cont.)	BRH$K_SUMMARY	Indicates that this record contains summary data describing the back-up as a whole.
	BRH$K_VOLUME	Indicates that this record contains the volume summary data used to reinitialize Files-11 volumes.
	BRH$K_FILE	Indicates a file attribute record, containing data describing a file.
	BRH$K_VBN	Indicates a file virtual block record, containing file contents.
	BRH$K_PHYSVOL	Indicates a physical volume attribute record.
	BRH$K_LBN	Indicates a physical volume logical block record.
	BRH$K_FID	Indicates a file identification record.
BRH$L_FLAGS	Contains the following flag bits for the record:	
	BRH$V_BADDATA	Set if a data error occurred while the data contained in this record was being read.
	BRH$V_DIRECTORY	Set if this record pertains to a directory file.
BRH$L_ADDRESS	If the record is a file virtual block record, contains the VBN in the file where the data in this record starts. If the record is a physical volume logical block record, this field contains the LBN where the data in this record starts. Otherwise, this field contains 0.	

B.5 ATTRIBUTE FORMAT

The attribute format records data items in summary, volume, file, and physical volume attribute records. Figure B-3 depicts the format for data items.

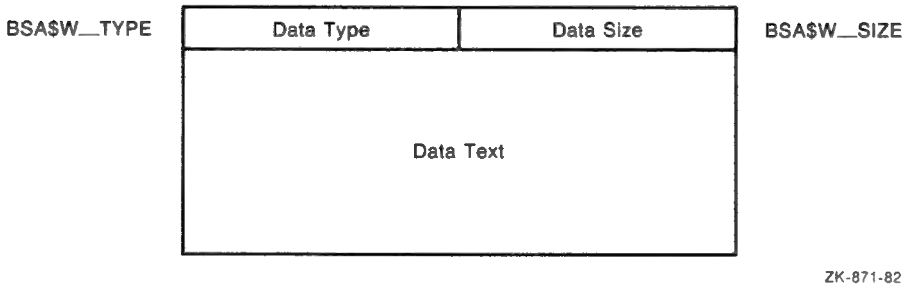


Figure B-3: Data Item

The first word gives the size of the data in bytes; the second word specifies the identity of the data item. The data type depends on the record type of the data (summary, volume, file, or physical volume). The remaining part is the data itself. Data items are packed into the record without any filler for alignment and can appear in any order.

B.6 BACKUP SUMMARY DATA TYPES

The BACKUP summary record contains data relevant to the back-up as a whole. It is the first data record in a save set. It is also the first data record in the first block of each volume of a multivolume save set. This allows each volume of the save set to be read and interpreted independently.

The first word of the summary record contains the structure level of the version of BACKUP that created the tape. For level 1, version 1, this value is 257. The remainder of the record contains data in attribute format. Table B-3 describes the summary data types.

Table B-3: Summary Data Types

Type Name	Maximum Size	Contents
BSA$K_SSNAME	32	Contains the save set file name.
BSA$K_COMMAND	512	Contains the command line that produced this save set.
BSA$K_COMMENT	512	Contains a user comment.
BSA$K_USERNAME	32	Contains the name of the user who wrote the BACKUP tape.
BSA$K_USERUIC	4	Contains the UIC of the user who wrote the BACKUP tape.
BSA$K_DATE	8	Contains the date and time at which the back-up took place.

(Continued on next page)

Table B-3 (Cont.): Summary Data Types

Type Name	Maximum Size	Contents
BSA$K_OPSYS	2	Contains the identification number of the operating system that performed the back-up. BACKUP writes the value 1024 in this field.
BSA$K_SYSVER	4	Contains the version of the operating system that performed the back-up.
BSA$K_NODENAME	12	Contains the node name of the operating system that performed the back-up.
BSA$K_SIR	4	Contains the CPU system identification register.
BSA$K_DRIVEID	16	Contains the device name of the drive that wrote the back-up.
BSA$K_BACKVER	32	Contains the version number of BACKUP that created the tape (in ASCII representation).
BSA$K_BLOCKSIZE	4	Contains the size (in bytes) of each block in the save set.
BSA$K_XORSIZE	2	Contains the number of blocks in each XOR group, not counting the XOR block.
BSA$K_BUFFERS	2	Contains the number of buffers used to write the BACKUP tape. This provides a limit on searching for a rewrite of a bad data block.
BSA$K_VOLSETNAM	12	Contains the volume set name.
BSA$K_NVOLS	2	Contains the number of volumes in the volume set.
BSA$K_BACKSIZE	8	Reserved.
BSA$K_BACKFILES	4	Reserved.

B.7 VOLUME SUMMARY RECORD DATA TYPES

The volume summary record contains the information necessary to initialize Files-11 disk volumes. Table B-4 describes the volume summary record data types.

Table B-4: Volume Summary Record Data Types

Type Name	Maximum Size	Contents
BSA$K_VOLSTRUCT	2	Contains the volume structure level.
BSA$K_VOLNAME	12	Contains the volume label.
BSA$K_OWNERNAME	12	Contains the name of the owner of the volume.
BSA$K_FORMAT	12	Contains the volume file format name.
BSA$K_RVN	2	Contains the relative volume number.
BSA$K_VOLOWNER	4	Contains the UIC of the owner of the volume.
BSA$K_PROTECT	2	Contains the volume protection mask.
BSA$K_FILEPROT	2	Contains the volume default file protection mask.
BSA$K_RECPROT	2	Contains the volume default record protection mask.
BSA$K_VOLCHAR	2	Contains the volume characteristics bit mask.
BSA$K_VOLDATE	8	Contains the volume creation date in 64-bit format.
BSA$K_WINDOW	1	Contains the default file window size.
BSA$K_LRU_LIM	1	Contains the default directory LRU limit.
BSA$K_EXTEND	2	Contains the default file extend size.
BSA$K_CLUSTER	2	Contains the storage bit map cluster factor.
BSA$K_RESFILES	2	Contains the number of reserved files on the volume.
BSA$K_VOLSIZE	3	Contains the original volume size in blocks.
BSA$K_TOTSIZE	8	Reserved.
BSA$K_TOTFILES	4	Reserved.
BSA$K_MAXFILES	4	Contains the maximum number of files allowed.
BSA$K_MAXFILNUM	4	Contains the highest file number in use on the volume.
BSA$K_SERIALNUM	4	Contains the volume serial number.

(Continued on next page)

Table B-4 (Cont.): Volume Summary Record Data Types

Type Name	Maximum Size	Contents
BSA$K_INDEXLBN	4	Contains the starting LBN of the index file bit map.
BSA$K_BOOTBLOCK	512	Contains the image contained in the boot block.
BSA$K_RETAINMIN	8	Contains the minimum retention period in 64-bit delta time format.
BSA$K_RETAINMAX	8	Contains the maximum retention period in 64-bit delta time format.

B.8 FILE ATTRIBUTE RECORD DATA TYPES

The file attribute record contains the information necessary to recreate files. The records containing data for each file follow the file attribute record. The first word of the file attribute record contains the structure level of the version of BACKUP that created the tape. For level 1, version 1, this value is 257. Table B-5 describes the file attribute data types.

Table B-5: File Attribute Record Data Types

Type Name	Maximum Size	Contents
BSA$K_FILENAME	128	Contains the name of the file, including its directory string, but not including the device name.
BSA$K_STRUCLEV	2	Contains the structure level of the volume that originally contained the file.
BSA$K_FID	6	Contains the file identification of the file, including relative volume number.
BSA$K_BACKLINK	6	Contains the file identification of the primary directory entry of the file.
BSA$K_FILESIZE	4	Contains the allocated size of the file in blocks.
BSA$K_UIC	4	Contains the owner UIC of the file.
BSA$K_FPRO	2	Contains the file protection mask for the file.
BSA$K_RPRO	2	Contains the record protection mask for the file.

(Continued on next page)

Table B-5 (Cont): File Attribute Record Data Types

Type Name	Maximum Size	Contents
BSA$K_ACLEVEL	1	Contains the access mode protection field of the file.
BSA$K_UCHAR	4	Contains the longword containing the file characteristics bits.
BSA$K_RECATTR	32	Contains the VAX-11 RMS record attributes area of the file.
BSA$K_REVISION	2	Contains the number of times the file has been revised.
BSA$K_CREDATE	8	Contains the file's creation date.
BSA$K_REVDATE	8	Contains the date the file was revised.
BSA$K_EXPDATE	8	Contains the file's expiration date.
BSA$K_BAKDATE	8	Contains the file's back-up date.
BSA$K_BOOTVBN	4	Contains the VBN within the file for the boot block.
BSA$K_PLACEMENT	2048	Contains placement data. The file placement data is described in Section B.11.
BSA$K_DIR_UIC	4	Contains the UIC of the parent directory.
BSA$K_DIR_FPRO	2	Contains the file protection of the parent directory.
BSA$K_DIR_STATUS	1	Contains the status of a directory, with respect to BACKUP. If the file is a directory, bits 0 through 3 represent the following: Bit 0 File is a directory file Bit 1 Directory was selected by BACKUP Bit 2 Directory was scanned by BACKUP Bit 3 Files in the directory were selected by BACKUP
BSA$K_DIR_VERLIM	2	Contains the default version limit of the parent directory.
BSA$K_VERLIMIT	2	Contains the version limit for the file.

B.9 PHYSICAL VOLUME ATTRIBUTE RECORD DATA TYPES

The physical volume attribute record contains information necessary to restore a save set created with the /PHYSICAL command qualifier. Table B-6 describes the physical volume attribute record data types.

Table B-6: Physical Volume Attribute Record Data Types

Type Name	Maximum Size	Contents
BSA$K_SECTORS	1	Contains the number of sectors per track.
BSA$K_TRACKS	1	Contains the number of tracks per cylinder.
BSA$K_CYLINDERS	2	Contains the number of cylinders per volume.
BSA$K_MAXBLOCK	4	Contains the number of logical blocks per volume.
BSA$K_DEVTYP	1	Contains the type of device that originally contained the data. The type is stored in code as defined by the $DCDEF macro.
BSA$K_SERIAL	4	Contains the serial number of the device that originally contained the data.
BSA$K_DEVNAM	64	Contains the device name of the device that originally contained the data.
BSA$K_LABEL	12	Contains the label of the device that originally contained the data.
BSA$K_BADBLOCK	2048	Contains the bad block descriptor for the device that originally contained the data. The data field contains a sequence of pairs of longwords. In each pair the first longword contains the starting LBN of a sequence of bad blocks, the second contains the number of blocks in the sequence.

B.10 FILE IDENTIFICATION RECORD DATA TYPES

The file identification record describes the status of files in a Files-11 volume. Table B-7 describes the File ID records.

Table B-7: File ID Record

Field Name	Contents
BSA$W_STRUCLEV	Contains the structure level of the version of BACKUP that wrote the File ID record. For level 1, version 1, the value is 257.
BSA$W_FID_NUM	Contains the NUM word of the file ID.
BSA$W_FID_RVN	Contains the RVN word of the file ID. The RVN word has two subfields: BSA$B_FID_RVN BSA$B_FID_NMX
BSA$W_FID_COUNT	Count of SEQ words contained in this record.
BSA$W_FID_SEQ	SEQ word of file ID, repeated BSA$W_FID_COUNT times.

B.11 FILE PLACEMENT DATA

The file placement data is contained in the file attribute record BSA$K_PLACEMENT. The file placement data contains data necessary to reconstruct the placement of an input file. Table B-8 describes the types of the file placement data; Table B-9 describes the offsets within the file placement data.

Table B-8: File Placement Data Types

Type Name	Maximum Size	Contents
BSA$K_PLC_FID	6	Contains the file identification.
BSA$K_PLC_COUNT	4	Contains the count of unplaced blocks.
BSA$K_PLC_PLACE	6	Contains the placement pointer (2 bytes) and the count of placed blocks (4 bytes).
BSA$K_PLC_PLLBN	10	Contains the placement pointer (2 bytes), the count of placed blocks (4 bytes), and the LBN of the placed blocks (4 bytes).

Table B-9: Offsets Into File Placement Data

Field Name	Contents
BSA$W_PLC_PTR	Contains the placement pointer.
BSA$L_PLC_COUNT	Contains the count of placed blocks.
BSA$L_PLC_LBN	Contains the LBN of placed blocks. This field is subdivided into two subfields: BSA$W_PLC_LOLBN Contains the low word of the LBN BSA$W_PLC_HILBN Contains the high word of the LBN

B.12 BACKUP JOURNAL FILE FORMATS

The BACKUP journal file contains a record of BACKUP operations and the saved files. The formats have been designed so that the journal file can be read starting at either end of the file. Each new BACKUP journal is appended, so it is possible to write an application that reads the most recent information first.

Each pair of records in the BACKUP journal file is separated by a byte containing the logical exclusive-or (XOR) of the lengths of the records on either side of the length byte. The length bytes on either end of a BACKUP journal contain the XOR of the first or last record and the value 0; thus the first and last length bytes contain the actual length of the first and last records in that BACKUP journal. The end of a BACKUP journal can be detected when the XOR of the length byte and the current record length produces a value of 0. The next journal begins with another count byte.

The first byte of each record indicates the BACKUP journal record type; records are variable length. The five BACKUP journal record types are listed in Table B-10.

Table B-10: BACKUP Journal Record Types

Record Type	Contents
BJL$K_STRUCLEV	Contains the structure level of the version of BACKUP that wrote the BACKUP journal file (two bytes). For level 1, version 1, the value is 257.
BJL$K_SSNAME	The save-set identification record contains the creation date of the save set (8 bytes) followed by the name of the save set (maximum of 32 bytes).
BJL$K_VOLUME	The volume identification record contains the volume label (12 bytes) followed by the relative volume number (2 bytes).

(Continued on next page)

Table B-10 (Cont.): BACKUP Journal Record Types

Record Type	Contents
BJL$K_DIRECTORY	The directory name record contains the XOR of the previous directory name and the next directory name (maximum of 79 bytes). If this record represents the first or last directory in the BACKUP journal file, it contains the XOR of the directory name and zero. The first data byte of this record contains the number of leading zeros that resulted from the XOR operation, which are not stored.
BJL$K_FILE	Contains a file name.

In order to make the BACKUP journal file readable in either direction, information is duplicated at the beginning and the end of the BACKUP journal file. Each BACKUP journal begins and ends with a structure level record followed (or preceded) by a save-set identification record. Each new save-set volume is bracketed by volume identification records. Each directory is bracketed by directory name records. The file name records are written in the order in which the files are saved.

B.13 SEQUENTIAL DISK FORMATS

When the Backup Utility writes a save set to a sequential disk volume, the save set is written using a subset of the Files-11 disk format. The save set file is contained in directory [000000] (the Master File Directory). The cluster factor of a sequential disk volume is 1.

If the save set fits on one disk volume, no volume set name is created and the relative volume number is zero. If the save set does not fit on one disk volume, a volume set name is created and the first volume is assigned a relative volume number of one.

Multivolume sequential disks are loosely coupled. The first volume does not contain the volume set count, because BACKUP has no way of determining the total number of volumes it will require. Similarly, the volume set list file ([0,0]VOLSET.SYS) contains nothing. If BACKUP initialized the first volume of a multivolume sequential disk, the part of the save set on that volume has a file identification of [10,1,1]; if the volume was not initialized, the file identification is unpredictable. The parts of the save set on any continuation volumes of a multivolume sequential save set have a file identification of [7,7,n].

APPENDIX C

FORMAT OF ACCOUNTING LOG FILE DATA

The following sections describe the format of the data records written to the accounting log file SYS$MANAGER:ACCOUNTNG.DAT.

These records are generated by a number of system events, including the following:

- Process or image deletion
- System initialization
- Login failure
- Printer jobs

User processes can send messages to the accounting log file using the Send Message to Accounting Manager ($SNDACC) system service. These messages can be used to write an arbitrary message to the accounting log file or to enable or disable accounting log file operations. The $SNDACC system service and the formats of these messages are documented in the VAX/VMS System Services Reference Manual.

The accounting record types, the offsets within the accounting records, and the other symbols used in these formats are all defined by the symbolic definition macro $ACRDEF.

NOTE

The formats described in this appendix are valid for VAX/VMS Version 3.0, but are subject to change without further advance notice at the time of any future system release.

C.1 FORMAT OF THE ACCOUNTING RECORD

An accounting record consists of an accounting record header and a number of information packets. The number of information packets depends on the type of information being sent.

Figure C-1 illustrates the general format of the accounting record; Table C-1 describes the fields contained in this record. The type field in the accounting record header (described in Table C-1) is subdivided into five fields. Table C-2 describes these fields.

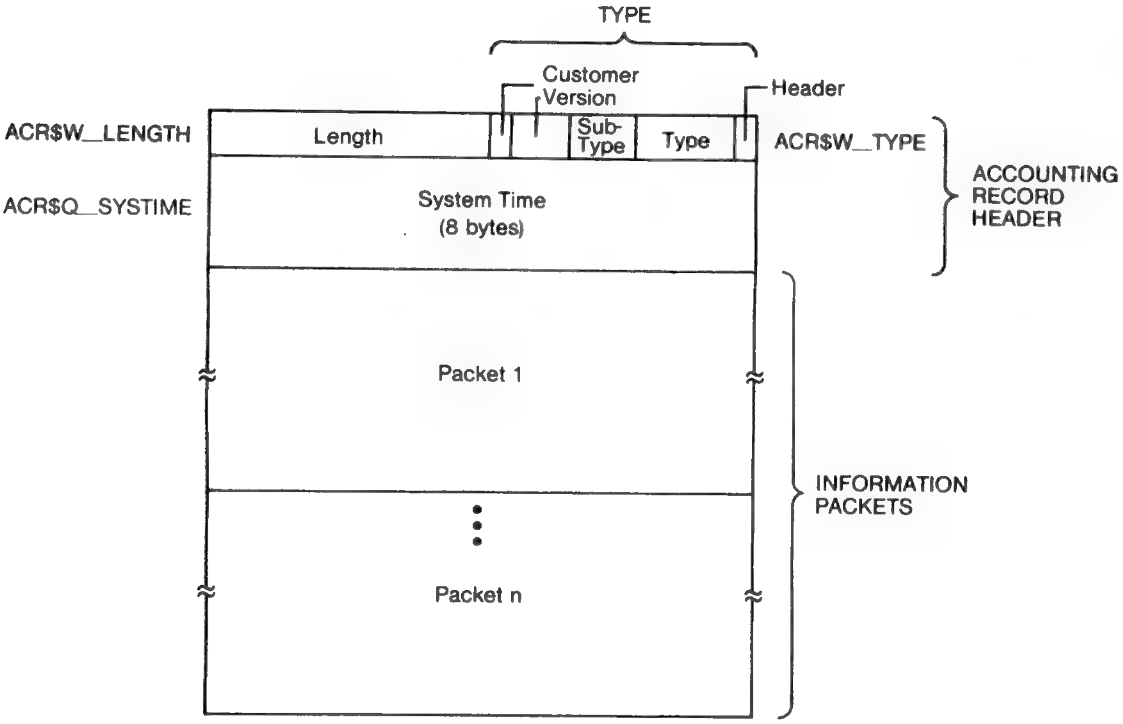


Figure C-1: Accounting Record Format

Table C-1: Descriptions of Accounting Record Fields

Field	Symbolic Offset	Contents
type	ACR$W_TYPE	Information describing the record. This field is subdivided into five fields, as described in Table C-2 (1 word)
length	ACR$W_LENGTH	Total length of the record (1 word)
system time	ACR$Q_SYSTIME	Current system time (1 quadword)
packet 1 - n		Information packets associated with the record (variable length)

Table C-2: Descriptions of ACR$W_TYPE Fields

Field	Symbolic Offset	Contents
header	ACR$V_PACKET	Identifies this header as a record header. This bit must be set to 0 (1 bit)

(continued on next page)

Table C-2 (Cont.): Descriptions of ACR$W_TYPE Fields

Field	Symbolic Offset	Contents
type	ACR$V_TYPE	Indicates the purpose of the record. There are currently 8 record types. Record types and their meanings are described in Section C.2 (7 bits)
subtype	ACR$V_SUBTYPE	Indicates the process type with which the record is associated. The subtypes are: **Symbol** — **Meaning** ACR$K_INTERACTIVE — Interactive Process ACR$K_SUBPROCESS — Subprocess ACR$K_DETACHED — Detached Process ACR$K_BATCH — Batch Process ACR$K_NETWORK — Network Process (4 bits)
version	ACR$V_VERSION	Indicates the accounting format with which the record is associated. The formats are: **Symbol** — **Meaning** ACR$K_VERSION2 — VAX/VMS Version 2.0 accounting format ACR$K_VERSION3T — VAX/VMS Version 3.0 field test accounting format ACR$K_VERSION3 — VAX/VMS Version 3.0 accounting format (3 bits)
customer	ACR$V_CUSTOMER	Identifies whether the record was written by DIGITAL software or by customer software. If the bit is not set, the record was written by DIGITAL software. If the bit is set to 1, the record was written by customer software (1 bit)

C.2 ACCOUNTING RECORD TYPES

Accounting record types identify the type of operation that caused the record to be sent. There are currently eight accounting record types.

Each type of accounting record requires a defined set of packets. Table C-3 describes the accounting record types and lists the packets required by each type. Note that the subtype field in the ACR$W_TYPE field is only meaningful when used with the first two record types: process deleted and image deleted.

Table C-3: Accounting Record Types

Symbol	Meaning
ACR$K_PRCDEL	A process was deleted. Requires: ACR$K_ID ACR$K_RESOURCE
ACR$K_IMGDEL	An image was deleted. Requires: ACR$K_ID ACR$K_RESOURCE ACR$K_IMAGENAME
ACR$K_SYSINIT	System was initialized. Requires: ACR$K_ID ACR$K_RESOURCE
ACR$K_LOGFAIL	A login validation failed. Requires: ACR$K_ID ACR$K_RESOURCE
ACR$K_PRINT	A print job was queued. Requires: ACR$K_ID ACR$K_PRINT
ACR$K_USER	User-supplied data. Requires: ACR$K_ID ACR$K_USER_DATA
ACR$K_FILE_FL	Accounting file forward link. Requires: ACR$K_FILENAME
ACR$K_FILE_BL	Accounting file backward link. Requires: ACR$K_FILENAME

C.3 ACCOUNTING PACKETS

There are six types of accounting packets:

- Identification Packet
- Resource Packet

- Image Name Packet
- Print Resource Packet
- File Name Packet
- User Data Packet

Section C.3.1 describes the general format of the accounting packets. Sections C.3.2 through C.3.7 describe the organization of the different accounting packets.

C.3.1 General Format of Accounting Packets

Each packet type contains a packet header, followed by data fields. The data fields can contain fixed-length data, variable-length data, or offsets to variable-length data. Offsets contain the distance, in bytes, from the beginning of the packet to the variable-length data.

All variable-length data are represented as counted strings. Variable-length data follow the last fixed-length data field in the packet. Figure C-2 illustrates the general format of the accounting packet.

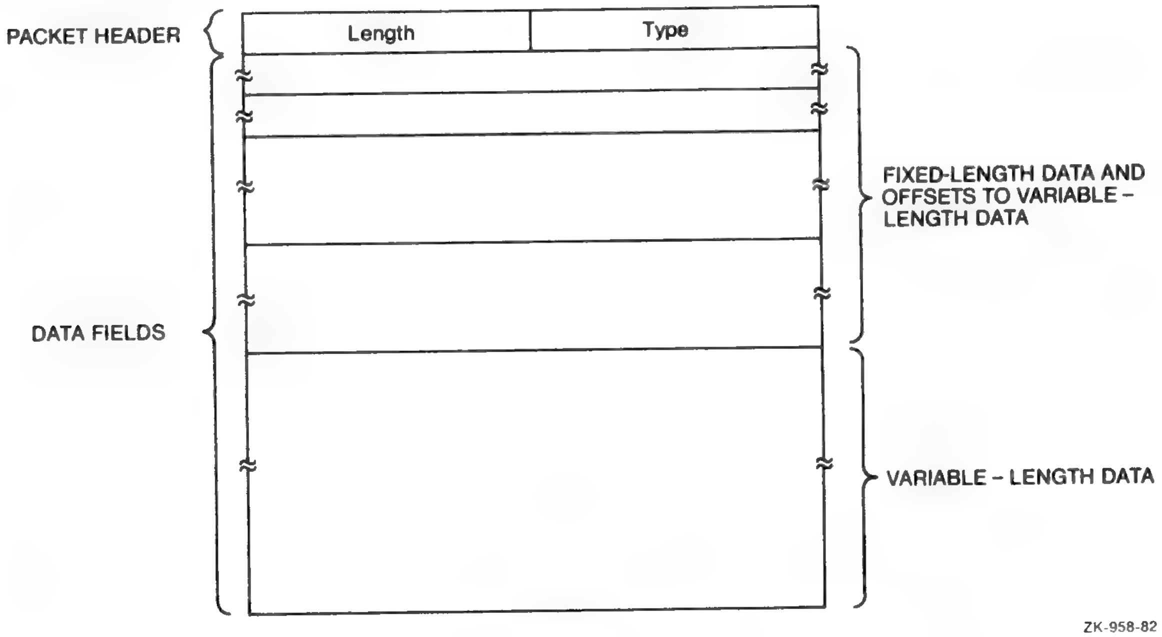


Figure C-2: Accounting Packet Format

Accounting packets need not use each type of data field. See Sections C.3.2 through C.3.7 for complete descriptions of the fields contained in each accounting packet.

All accounting packets start with a packet header. The packet header uses the same symbolic offsets as the first longword of the record header. Figure C-3 illustrates the accounting packet header; Table C-4 describes the fields in this header. The type field in the accounting packet header is subdivided into five fields. Table C-5 describes these fields.

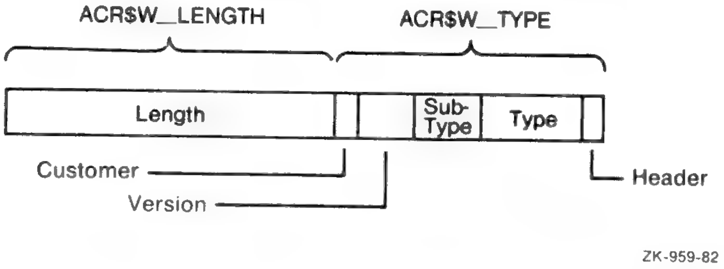


Figure C-3: Accounting Packet Header Format

Table C-4: Descriptions of Accounting Packet Header Fields

Field	Symbolic Offset	Contents
type	ACR$W_TYPE	Information describing the packet (1 word). This field is subdivided into five fields, as described in Table C-5
length	ACR$W_LENGTH	Total length of the packet. (1 word)

Table C-5: Descriptions of ACR$W_TYPE Fields

Field	Symbolic Offset	Contents
header	ACR$V_PACKET	Identifies this header as a packet header. This bit must be set to 1 (1 bit)
type	ACR$V_TYPE	Indicates the purpose of the packet. There are currently six packet types. These packet types are described in Sections C.3.2 through C.3.7 (7 bits)
subtype	ACR$V_SUBTYPE	Indicates the packet subtype; reserved for future use (4 bits)

(continued on next page)

Table C-5 (Cont.): Descriptions of ACR$W_TYPE Fields

Field	Symbolic Offset	Contents
version	ACR$V_VERSION	Indicates the accounting format with which the record is associated. The formats are: Symbol — Meaning ACR$K_VERSION2 — VMS Version 2.0 accounting format ACR$K_VERSION3T — VMS Version 3.0 field test accounting format ACR$K_VERSION3 — VMS Version 3.0 accounting format (3 bits)
customer	ACR$V_CUSTOMER	Identifies whether the record was written by DIGITAL software or by customer software. If the bit is not set, the record was written by DIGITAL software. If the bit is set to 1, the record was written by customer software. (1 bit)

C.3.2 Packet Type ACR$K_ID (Identification Packet)

The identification packet identifies the process that caused information to be sent to the accounting manager.

Figure C-4 depicts the organization of the identification packet; Table C-6 describes the fields contained in the packet. See Section C.3.1 for more information on the packet header.

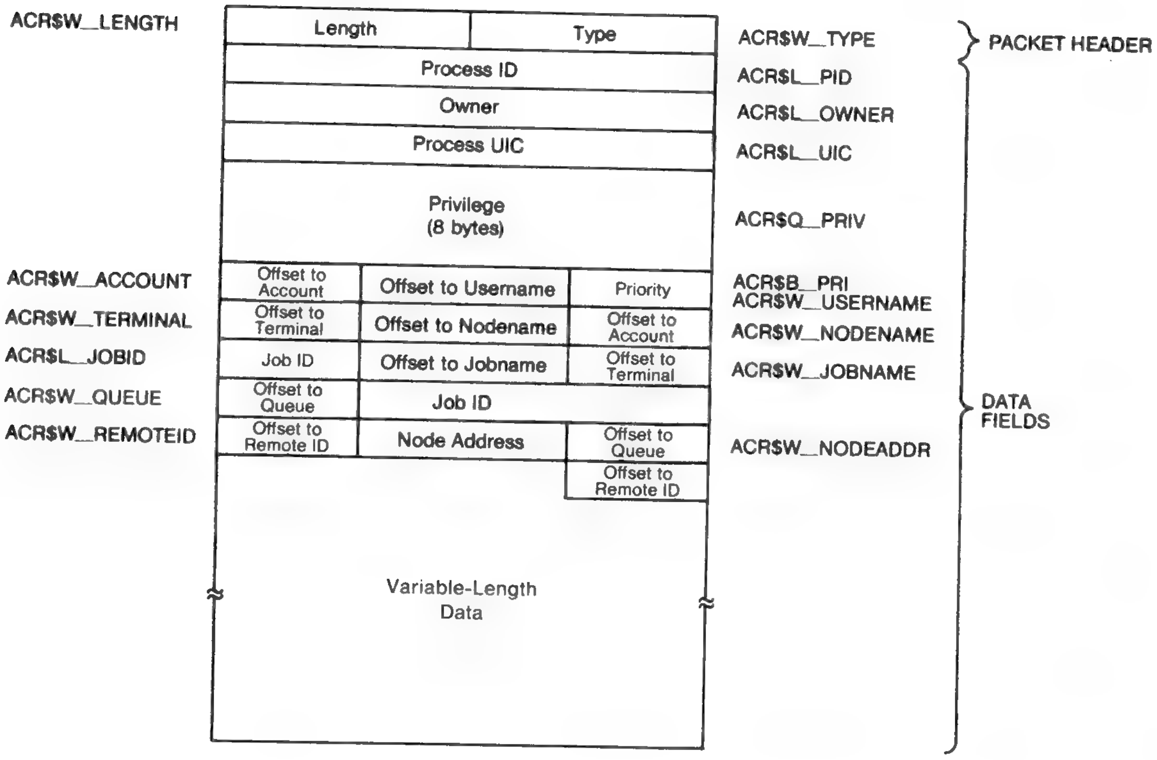


Figure C-4: Block Diagram for ACR$K_ID

Table C-6: Field descriptions for ACR$K_ID

Field	Symbolic Offset	Contents
pid	ACR$L_PID	Process identification of the process (longword)
owner	ACR$L_OWNER	Process identification of owner process, if the process is a subprocess. If the process is not a subprocess, the value is 0 (longword)
uic	ACR$L_UIC	Process UIC of the process. The UIC can be addressed as two separate words: ACR$W_MEM for the member number, and ACR$W_GRP for the group number (longword)
privilege	ACR$Q_PRIV	Privileges held by the process (quadword)
prio	ACR$B_PRI	Priority of the process (byte)

(continued on next page)

Table C-6 (Cont.): Field descriptions for ACR$K_ID

Field	Symbolic Offset	Contents
username	ACR$W_USERNAME	Offset to counted ASCII string containing the user name of the process (word)
account	ACR$W_ACCOUNT	Offset to counted ASCII string containing the account name of the process (word)
node name	ACR$W_NODENAME	Offset to counted ASCII string containing the node name of the remote process (word)
terminal	ACR$W_TERMINAL	Offset to counted ASCII string containing the terminal name of the process (word)
jobname	ACR$W_JOBNAME	Offset to counted ASCII string containing the job name of the process (word)
jobid	ACR$L_JOBID	Identification of the job (longword)
queue	ACR$W_QUEUE	Offset to counted ASCII string containing the name of the queue with which a batch or print job is associated (word)
node address	ACR$W_NODEADDR	Contains the remote node address (word)
remote id	ACR$W_REMOTEID	Offset to counted ASCII string containing the remote ID of the remote process (varies with network implementation and use) (word)

C.3.3 Packet Type ACR$K_RESOURCE (Resource Packet)

The resource packet contains information on the identified process.

Figure C-5 depicts the organization of the resource packet; Table C-7 describes the fields contained in the packet. See Section C.3.1 for more information on the packet header.

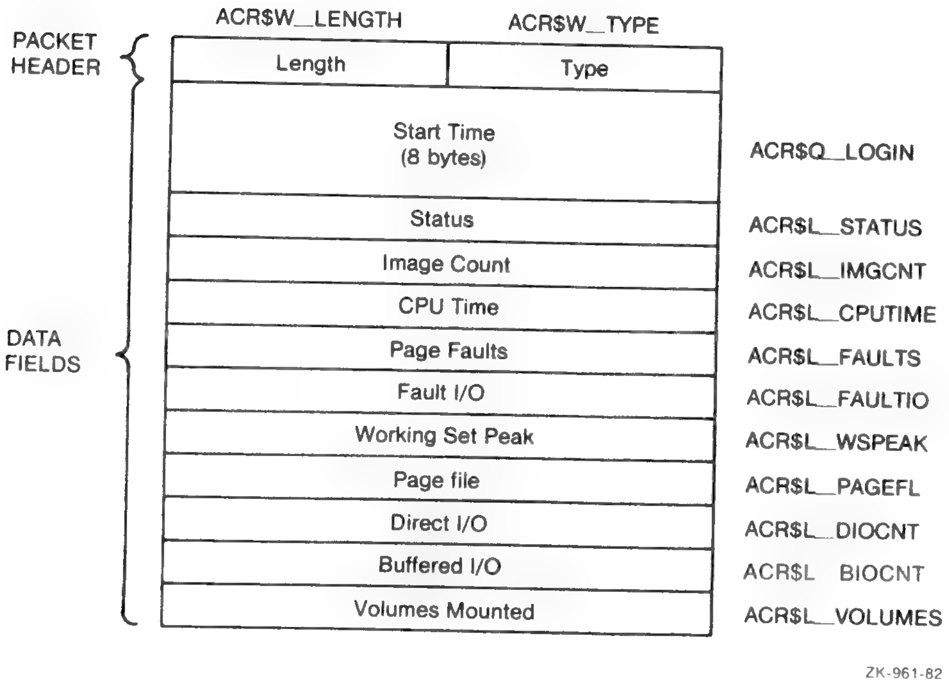


Figure C-5: Block Diagram for ACR$K_RESOURCE

Table C-7: Field descriptions for ACR$K_RESOURCE

Field	Symbolic Offset	Contents
start time	ACR$Q_LOGIN	64-bit binary time at which process or image was started (quadword)
status	ACR$L_STATUS	Final status of the process or image (longword)
image count	ACR$L_IMGCNT	Execution count or sequence number of the image (longword)
cputime	ACR$L_CPUTIME	Total CPU time of the process or image (longword)
page faults	ACR$L_FAULTS	Page fault count of the process or image (longword)
fault I/O	ACR$L_FAULTIO	Page fault I/O count of the process or image (longword)
wspeak	ACR$L_WSPEAK	Peak working set of the process or image (longword)
pagefile	ACR$L_PAGEFL	Peak page file usage of the process or image (longword)

(continued on next page)

Table C-7 (Cont.): Field descriptions for ACR$K_RESOURCE

Field	Symbolic Offset	Contents
dir I/O	ACR$L_DIOCNT	Direct I/O count of the process or image (longword)
buff I/O	ACR$L_BIOCNT	Buffered I/O count of the process or image (longword)
vol mount	ACR$L_VOLUMES	Number of volumes mounted by the process or image (longword)

C.3.4 Packet Type ACR$K_IMAGENAME (Image Name Packet)

The image name packet contains the name of image run by the identified process.

Figure C-6 depicts the organization of the image name packet; Table C-8 describes the fields contained in the packet. See Section C.3.1 for more information on the packet header.

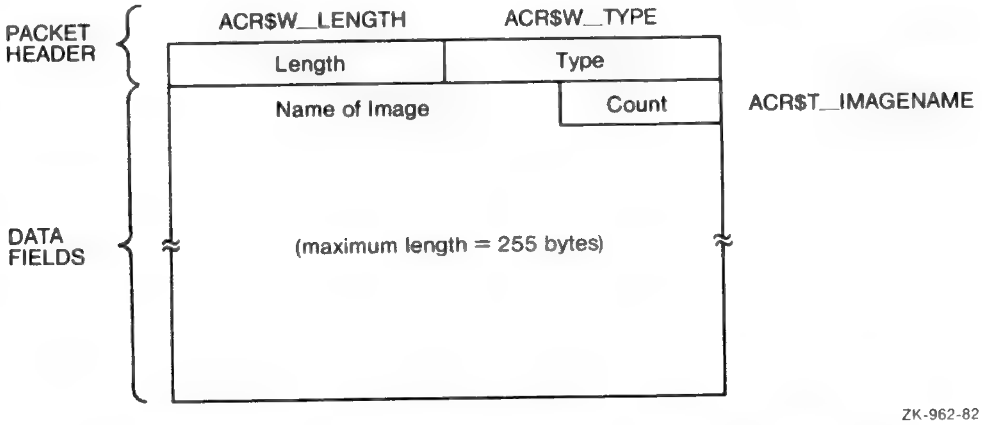


Figure C-6: Block Diagram for ACR$K_IMAGENAME

Table C-8: Field descriptions for ACR$K_IMAGENAME

Field	Symbolic Offset	Contents
imagename	ACR$T_IMAGENAME	Name of the image (counted ASCII string)

C.3.5 Packet Type ACR$K_PRINT (Print Resource Packet)

The print resource packet contains information on printer jobs.

Figure C-7 depicts the organization of the print resource packet; Table C-9 describes the fields contained in the packet. See Section C.3.1 for more information on the packet header.

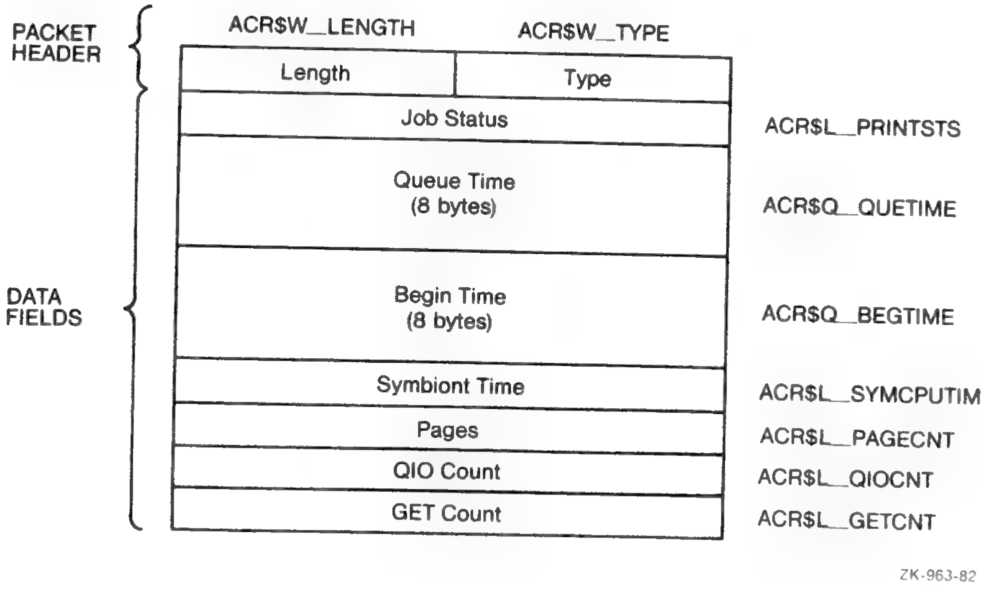


Figure C-7: Block Diagram for ACR$K_PRINT

Table C-9: Field descriptions for ACR$K_PRINT

Field	Symbolic Offset	Contents
job status	ACR$L_PRINTSTS	Status of the printer job (longword)
queue time	ACR$Q_QUETIME	Time the job was queued (64-bit binary time)
begin time	ACR$Q_BEGTIME	Time the job was started (64-bit binary time)
symb time	ACR$L_SYMCPUTIM	Total symbiont CPU time (longword)
pages	ACR$L_PAGECNT	Total number of pages printed (longword)
qio count	ACR$L_QIOCNT	Total number of QIOs issued (longword)
get count	ACR$L_GETCNT	Total number of GETs issued (longword)

C.3.6 Packet Type ACR$K_FILENAME (File Name Packet)

The file name packet contains the name of the accounting file to point to or modify. Figure C-8 depicts the organization of the file name packet; Table C-10 describes the fields contained in the packet. See Section C.3.1 for more information on the packet header.

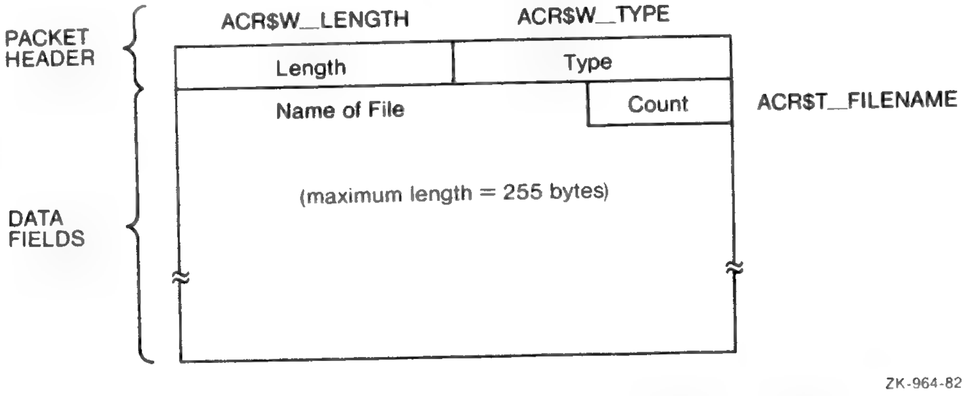


Figure C-8: Block Diagram for ACR$K_FILENAME

Table C-10: Field descriptions for ACR$K_FILENAME

Field	Symbolic Offset	Contents
file name	ACR$T_FILENAME	Name of the file (counted ASCII string)

C.3.7 Packet Type ACR$K_USER_DATA (User Data Packet)

The user data packet contains user information to be sent to the accounting manager. Figure C-9 depicts the organization of the user data packet; Table C-11 describes the fields contained in the packet. See Section C.3.1 for more information on the packet header.

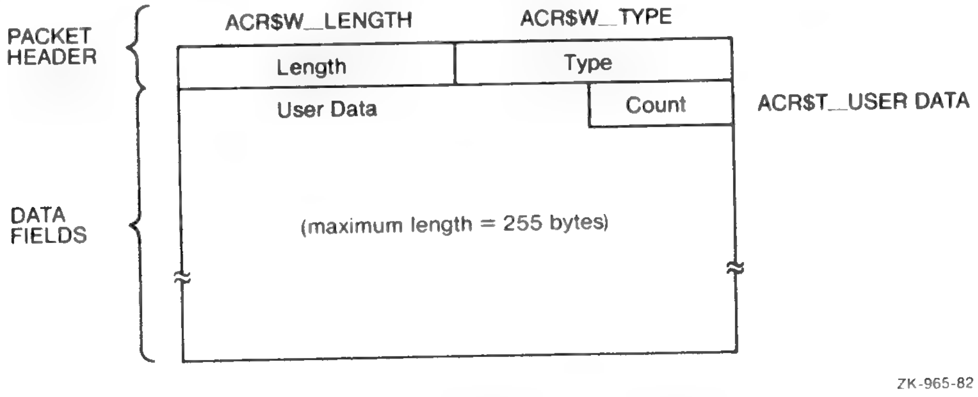


Figure C-9: Block Diagram for ACR$K_USER_DATA

Table C-11: Field descriptions for ACR$K_USER_DATA

Field	Symbolic Offset	Contents
user data	ACR$T_USER_DATA	Any user data (counted string)

APPENDIX D

MONITOR UTILITY RECORDING FILE RECORD FORMATS

The MONITOR recording file is the file into which binary performance data is written when a MONITOR request indicates "recording." A record is written to this file once for each requested class per interval; the record contains a predefined set of data for each of the requested performance classes.

The MONITOR recording file is created when a MONITOR request is initiated, and is closed when the request terminates. The MONITOR recording file may be used as a source file to format and display the data on a terminal, to create a summary file, or to record a new recording file with different characteristics.

NOTE

> The record formats described in this appendix are subject to change without notice at any future major VAX/VMS release.

D.1 FORMAT OF MONITOR RECORDING FILE

The MONITOR recording file is a VAX-11 RMS sequential file with variable-length records. Each record in the file begins with a one-byte type field. The remaining fields are different in length and format for each record type. There are three categories of record types:

- Customer control record
- DIGITAL control record
- Class record

Customer control records may appear anywhere in the recording file. They are not generated by MONITOR and are ignored by MONITOR when it reads the file.

The first records in the MONITOR recording file, excluding customer control records, are DIGITAL control records. There are currently two types of DIGITAL control records: the file header record and the system information record. These are, respectively, the first two records in the MONITOR recording file.

The class records, which contain data on requested performance classes, follow the DIGITAL control records. The class record is generally written once per interval for each class being recorded. An exception to this rule is the case where several class records are required to contain data for a single class over a single interval. This case can occur for the PROCESSES class when there are too many processes to be accommodated by the maximum record size.

Unique record types are assigned to each MONITOR record. Record types 0-127 are reserved for class records; record types 128-191 are reserved for DIGITAL control records; record types 192-255 are reserved for customer control records.

There are currently eleven record types generated by the Monitor Utility. Table D-1 lists the MONITOR record types.

Table D-1: MONITOR Record Types

Type	Type Number (in decimal)
File Header	128
System Information	129
PROCESSES Class	0
STATES Class	1
MODES Class	2
PAGE Class	3
IO Class	4
FCP Class	5
POOL Class	6
LOCK Class	7
DECNET Class	8

D.2 CONVENTIONS

The following sections define the contents of each field within each record type. Record type and record size are given in decimal representation. References to "system time" indicate time values in system time format, that is, in 64-bit format.

The field offset names listed are not defined within the Monitor Utility. However, DIGITAL recommends that you define and use these offset names when you work with MONITOR output records.

The suggested naming convention for the field offset names appears below. Each name is of the form:

```
MNR_CCC$X_DDDDD
```

CCC is a record type or class mnemonic

X is a one-letter code indicating the size of the data item as follows:

B for byte
W for word
L for longword
Q for quadword
O for octaword
T for ASCII string

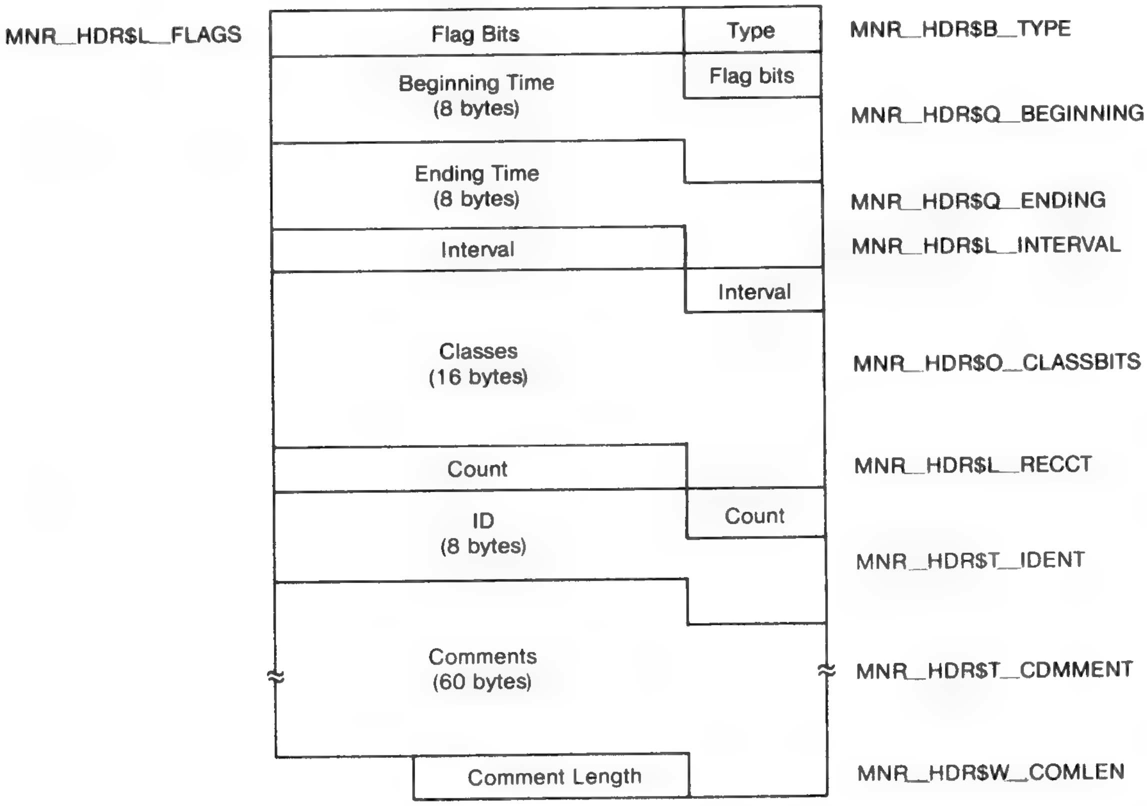
DDDDD is the name describing the data item.

In the following tables, the size of the data is also shown in parentheses, following the description of the field contents.

D.3 FILE HEADER RECORD

There is one file header record per file. It contains information applicable to all classes of performance data contained in the file. It must be the first record (except for customer control records) in the file.

The file header record has a record type of 128 and a size of 115 bytes. Figure D-1 illustrates the format of the file header record; Table D-2 describes the fields in this record.



ZK-966-82

Figure D-1: File Header Record Format

Table D-2: Descriptions of File Header Record Fields

Field	Symbolic Offset	Contents
type	MNR_HDR$B_TYPE	Record type identifier (1 byte)
flags	MNR_HDR$L_FLAGS	32 flag bits; low-order bit = bit 0. All flags reserved to DIGITAL for future use. (1 longword)
beginning	MNR_HDR$Q_BEGINNING	System time of beginning of recording (1 quadword)
ending	MNR_HDR$Q_ENDING	System time of end of recording (1 quadword)
interval	MNR_HDR$L_INTERVAL	Interval in seconds between collections; this is the value specified by the user in the recording request. It is not necessarily equal to the exact interval value obtained by subtracting two consecutive time stamps for a given class (1 longword)
classes	MNR_HDR$O_CLASSBITS	128-bit string representing all classes; a bit set to 1 indicates the presence in this file of the class whose type number corresponds to the bit number. Low-order bit = bit 0 (1 octaword)
count	MNR_HDR$L_RECCT	Count of all records in the file (1 longword)
ID	MNR_HDR$T_IDENT	MONITOR Recording File Structure Level Identification (MONSL001) (8 bytes)
comment	MNR_HDR$T_COMMENT	Recording file description supplied by the user, including trailing blanks (60 bytes)
comment length	MNR_HDR$W_COMLEN	Actual length of recording file description string specified by the user (1 word)

D.4 SYSTEM INFORMATION RECORD

There is one system information record per file. It contains information about the VAX/VMS system being monitored. It must be the second record (except for customer control records) in the file.

The system information record has a record type of 129 and a size of 14 bytes. Figure D-2 illustrates the format of the file header record; Table D-3 describes the fields in this record.

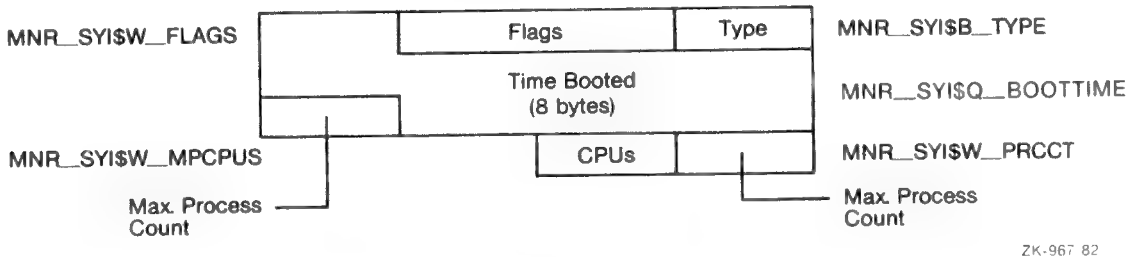


Figure D-2: System Information Record Format

Table D-3: Descriptions of System Information Record Fields

Field	Symbolic Offset	Contents
type	MNR_SYI$B_TYPE	Type identifier (1 byte)
flags	MNR_SYI$W_FLAGS	16 flag bits; low-order bit = bit 0. All flags reserved to DIGITAL for future use. (1 word)
time booted	MNR_SYI$Q_BOOTTIME	System time at which system booted (1 quadword)
max process count	MNR_SYI$W_MAXPRCCT	MAXPROCESSCNT system parameter value (1 word)
CPUs	MNR_SYI$B_MPCPUS	Number of CPUs. Contains a value of 2 when the system is a VAX-11/782 attached processor system; otherwise, contains a value of 1 (1 byte)

D.5 CLASS RECORDS

The MONITOR recording file contains one class record for each requested class per collection interval, except for the PROCESSES class. (See Section D.5.2 for more information on the PROCESSES class records.) For example, if a MONITOR user requested to record five classes (excluding PROCESSES) for a duration of 100 collection intervals, the file would contain 500 class records. Class records occur in order of increasing type number within an interval. The first class record for a given interval follows the last class record for the previous interval.

D.5.1 General Format

For all classes except PROCESSES, a class record consists of a class header followed by a data block. The PROCESSES class record has a class header, a class prefix, and one data block per process. Figure D-3 illustrates the format for class records.

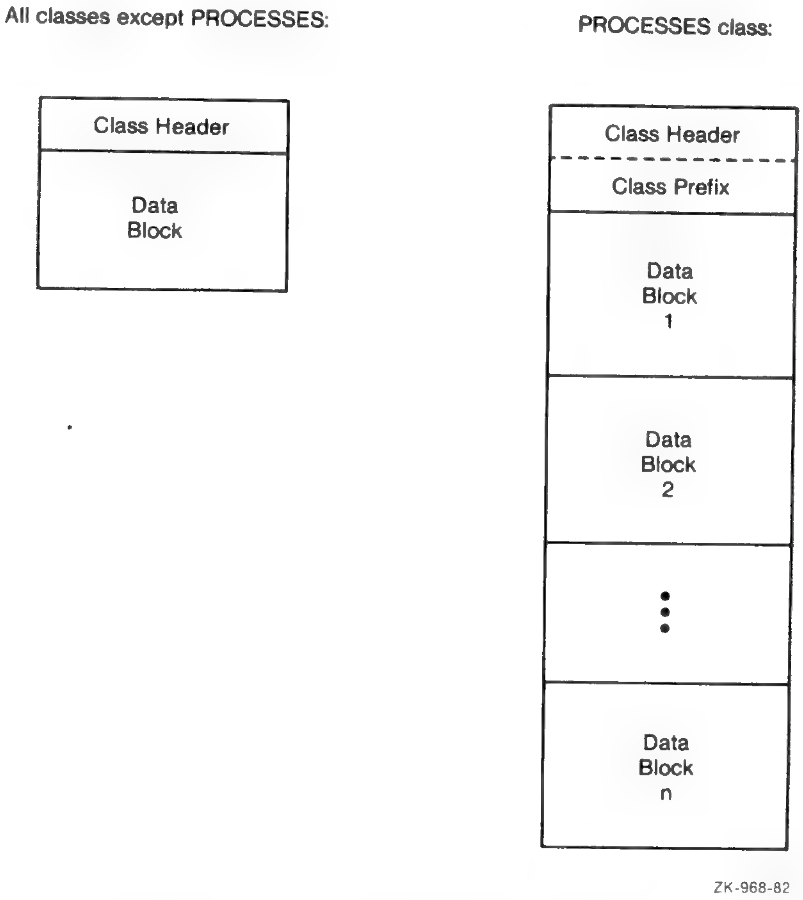


Figure D-3: Class Record Format

D.5.1.1 Class Header - The class header is the first part of every class record. Its format is independent of class. The class header is 13 bytes long.

Figure D-4 illustrates the format of the class header; Table D-4 describes the fields in the class header.

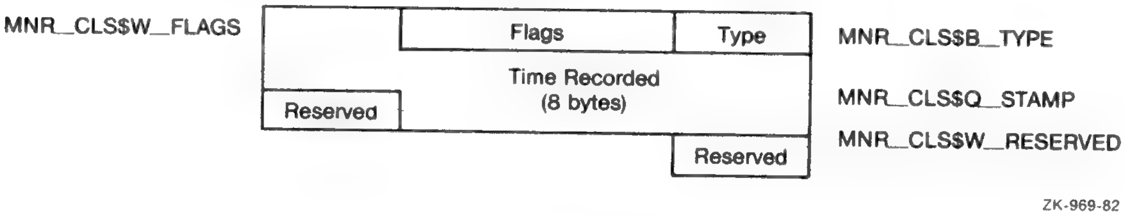


Figure D-4: Class Header Format

Table D-4: Descriptions of Class Header Fields

Field	Symbolic Offset	Contents
type	MNR_CLS$B_TYPE	Record type identifier (1 byte)
flags	MNR_CLS$W_FLAGS	16 flag bits; low order bit = bit 0. If bit 0 is set to 1, the data for this interval continues in the next record. Can be set for the PROCESSES class only. All other flags reserved by DIGITAL for future use (1 word)
time	MNR_CLS$Q_STAMP	System time at which this class record was recorded. The time value is non-decreasing across all class records in the file
reserved	MNR_CLS$W_RESERVED	Reserved for DIGITAL use (1 word)

D.5.1.2 Class Prefix (PROCESSES Class Only) - The class prefix always follows the class header for PROCESSES class records. It contains data describing the number of processes represented by the class record(s) for the current collection interval. Unlike other class records, that have one data block per record, the PROCESSES class has one data block per process. One of the class prefix data items describes the number of processes (and therefore the number of data blocks) included in the class record. The other class prefix data item describes the number of processes included in the interval.

It is possible to monitor a sufficiently large number of processes such that the required number of data blocks for one collection interval will not fit into a single maximum size record. In this case, the required number of PROCESSES class records is created to fully describe the processes.

All class headers in the set of PROCESSES class records for a given interval are identical, except for the setting of Bit 0 in the MNR_CLS$W_FLAGS field. This bit is set to 1 for all records except the last, for which it is set to 0.

The class prefixes in the set of class records vary, as described in Table D-5. The contents of the MNR_PRO$L_PCTREC field depend on the number of data blocks contained in the record; the contents of the MNR_PRO$L_PCTINT field remain constant for each record in the set. All records in the set except the last contain as many data blocks as will fit into the maximum size record (32000 bytes). The last record in the set contains the remaining data blocks.

Figure D-5 illustrates the class prefix format; Table D-5 describes the fields in the class prefix. The class prefix is 8 bytes long.

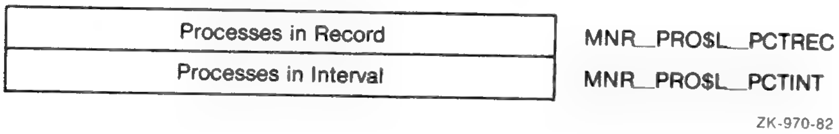


Figure D-5: Class Prefix Format

Table D-5: Descriptions of Class Prefix Fields

Field	Symbolic Offset	Contents
processes in record	MNR_PRO$L_PCTREC	Count of processes (data blocks) in this record (1 longword)
processes in interval	MNR_PRO$L_PCTINT	Count of processes (data blocks) for this interval (1 longword)

D.5.1.3 Data Block - The size and format of each data block and the number of blocks per record depend on the class. All classes except PROCESSES have one data block per record. The PROCESSES class has one data block per process. The fields within each block are performance data items.

The following sections describe the data items within the data block for each class. Every data item falls into one of three categories. It is either a count, a level, or an informational item. A count is a numeric quantity that increases at each succeeding interval for the life of a system boot. A level is a numeric quantity that may increase or decrease at each succeeding interval. An informational item represents data that, rather than being a unit of performance measurement (as are the first two types), is descriptive in nature.

In the tables that follow, item types are identified by the letters C (count), L (level) and I (informational). The item types are shown in parentheses, following the length of the field.

D.5.2 PROCESSES Class Record

The PROCESSES class record contains data describing all processes in the system. The PROCESSES class record has a record type of 0; its size depends on the number of processes being monitored. The size, in bytes, is calculated by adding the size of the class header, the class prefix, and the data blocks contained in the record. This is shown by the following formula:

```
21 + (51 * the value of MNR_PRO$L_PCTREC)
```

Figure D-6 illustrates the format of the PROCESSES class record. Table D-6 describes the fields in the data block for the PROCESSES class record. See Sections D.5.1.1 and D.5.1.2 for more information on the class header and the class prefix.

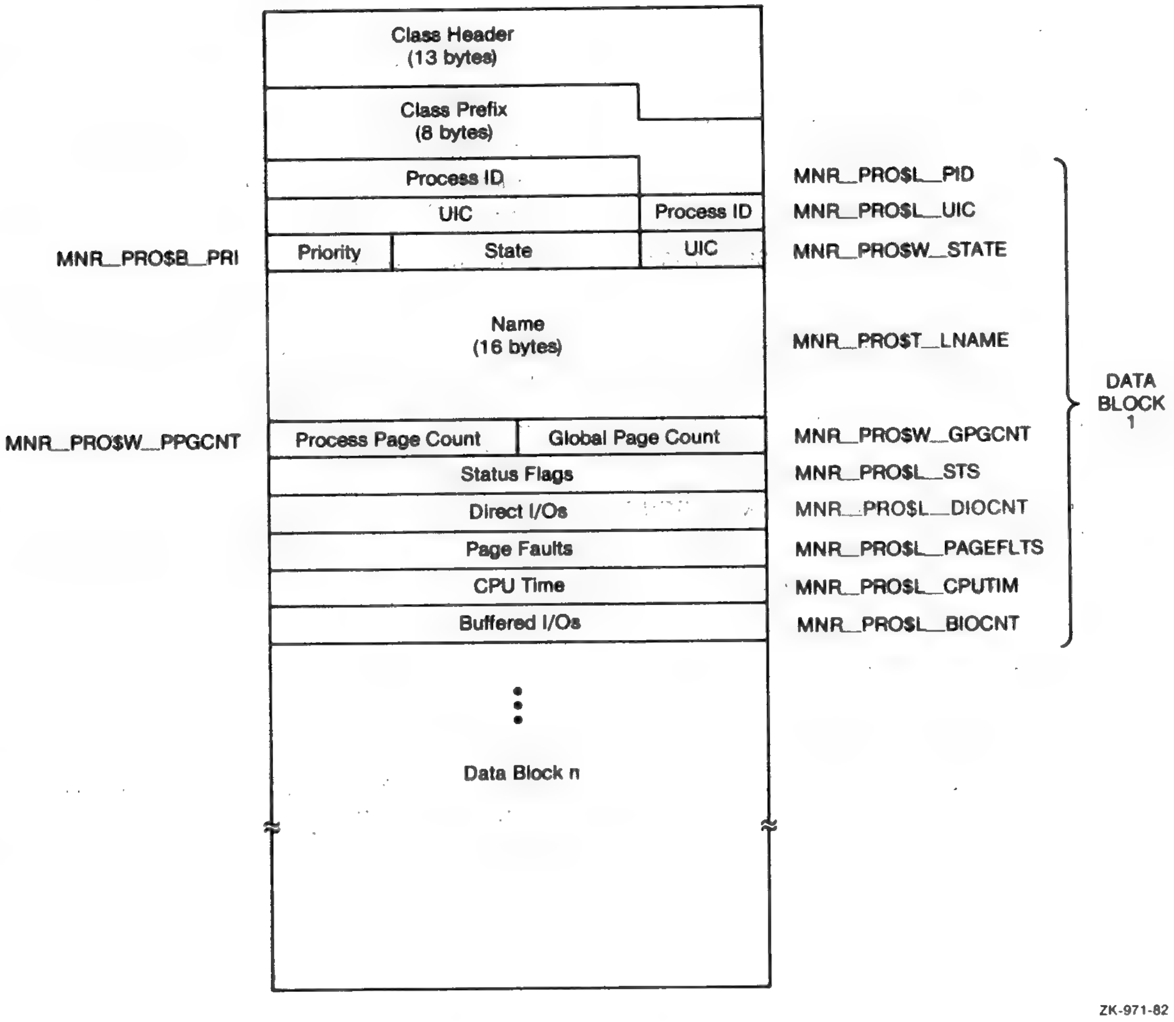


Figure D-6: PROCESSES Class Record Format

Table D-6: Descriptions of PROCESSES Class Record Fields

Field	Symbolic Offset	Contents
process ID	MNR_PRO$L_PID	Process identification (longword,I)
UIC	MNR_PRO$L_UIC	User identification code (Group is high-order word; Member is low-order word) (longword,I)
state	MNR_PRO$W_STATE	Current scheduling state code (word,I)
priority	MNR_PRO$B_PRI	Current software priority (complement of 31) (byte,I)
name	MNR_PRO$T_LNAME	Process name (counted ASCII string) (16 bytes,I)
global page count	MNR_PRO$W_GPGCNT	Current global page count (word,L)
process page count	MNR_PRO$W_PPGCNT	Current process page count (word,L)
status flags	MNR_PRO$L_STS	Software process status flags (PCB$V_RES bit clear implies swapped out) (longword,I)
Direct I/Os	MNR_PRO$L_DIOCNT	Direct I/O count (0 if swapped out) (longword,C)
page faults	MNR_PRO$L_PAGEFLTS	Page fault count (0 if swapped out) (longword,C)
CPU time	MNR_PRO$L_CPUTIM	Accumulated CPU time, in 10ms ticks (0 if swapped out) (longword,C)
Buffered I/Os	MNR_PRO$L_BIOCNT	Buffered I/O count (0 if swapped out) (longword,C)

D.5.3 STATES Class Record

The STATES class record contains data describing the number of processes in each of the scheduler states. The STATES class record has a record type of 1 and a size of 69 bytes.

Figure D-7 illustrates the format of the STATES class record. Table D-7 describes the fields in the data block for the STATES class record. See Section D.5.1.1 for more information on the class header.

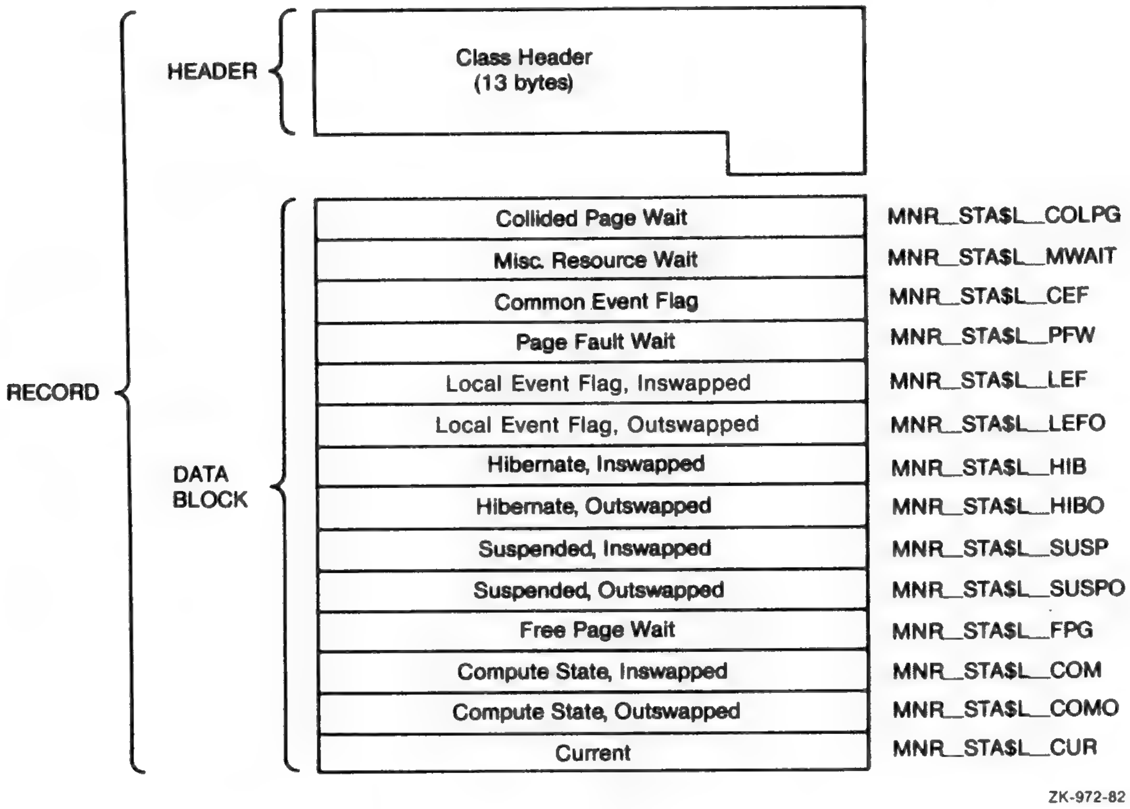


Figure D-7: STATES Class Record Format

Table D-7: Descriptions of STATES Class Record Fields

Field	Symbolic Offset	Contents
collided page wait	MNR_STA$L_COLPG	Number of processes in collided page wait (longword,L)
misc resource wait	MNR_STA$L_MWAIT	Number of processes in miscellaneous resource wait (longword,L)
common event flag wait	MNR_STA$L_CEF	Number of processes in common event flag wait (longword,L)
page fault wait	MNR_STA$L_PFW	Number of processes in page fault wait (longword,L)
loc event flag, insw	MNR_STA$L_LEF	Number of processes in local event flag wait, inswapped (longword,L)
loc event flag, outsw	MNR_STA$L_LEFO	Number of processes in local event flag wait, outswapped (longword,L)
hibernate inswapped	MNR_STA$L_HIB	Number of processes in hibernate wait, inswapped (longword,L)
hibernate outswapped	MNR_STA$L_HIBO	Number of processes in hibernate hibernate wait, outswapped (longword,L)
suspended inswapped	MNR_STA$L_SUSP	Number of processes in suspended wait, inswapped (longword,L)
suspended outswapped	MNR_STA$L_SUSPO	Number of processes in suspended wait, outswapped (longword,L)
free page wait	MNR_STA$L_FPG	Number of processes in free page wait (longword,L)
compute state inswapped	MNR_STA$L_COM	Number of processes in compute state, inswapped (longword,L)
compute state outswapped	MNR_STA$L_COMO	Number of processes in compute state, outswapped (longword,L)
current	MNR_STA$L_CUR	Number of current processes (longword,L)

D.5.4 MODES Class Record

The MODES class record contains data describing time spent in each of the processor modes. The MODES class record has a record type of 2. Its size is 41 bytes when monitoring a single processor, and 69 bytes when monitoring two processors (VAX-11/782).

Figure D-8 illustrates the format of the MODES class record. Table D-8 describes the fields in the data block for the MODES class record. The first seven fields refer to the primary processor. The last seven fields refer to the attached processor; they are included in the MODES record only if the MNR_SYI$B_MPCPUS field of the System Information Record contains a value of 2, indicating that two processors are being monitored.

See Section D.5.1.1 for more information on the class header.

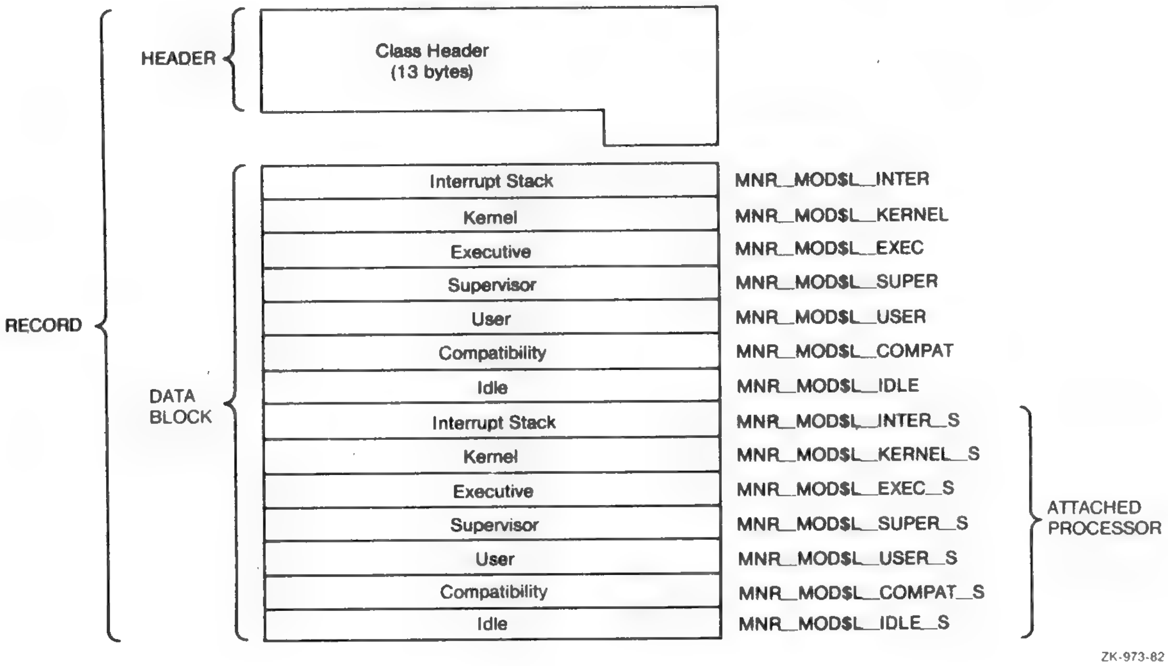


Figure D-8: MODES Class Record Format

Table D-8: Descriptions of MODES Class Record Fields

Fields that refer to the primary processor:		
Field	Symbolic Offset	Contents
interrupt stack	MNR_MOD$L_INTER	Count of clock ticks (10-millisecond units) spent on interrupt stack since system was booted (longword,C)
kernel	MNR_MOD$L_KERNEL	Count of clock ticks since system boot spent in kernel mode, excluding interrupt stack time (longword,C)
executive	MNR_MOD$L_EXEC	Count of clock ticks since system boot spent in exec mode (longword,C)
supervisor	MNR_MOD$L_SUPER	Count of clock ticks since system boot spent in supervisor mode (longword,C)
user	MNR_MOD$L_USER	Count of clock ticks since system boot spent in user mode, excluding compatibility-mode time (longword,C)
compatibility	MNR_MOD$L_COMPAT	Count of clock ticks since system boot spent in compatibility mode (longword,C)
idle	MNR_MOD$L_IDLE	Count of clock ticks since system boot spent executing the NULL process (longword,C)
Additional fields that refer to the attached processor:		
Field	Symbolic Offset	Contents
interrupt stack	MNR_MOD$L_INTER_S	Count of clock ticks (10-millisecond units) spent on interrupt stack since system was booted (longword,C)
kernel	MNR_MOD$L_KERNEL_S	Count of clock ticks since system boot spent in kernel mode, excluding interrupt stack time (longword,C)
executive	MNR_MOD$L_EXEC_S	Count of clock ticks since system boot spent in exec mode (longword,C)
supervisor	MNR_MOD$L_SUPER_S	Count of clock ticks since system boot spent in supervisor mode (longword,C)

(continued on next page)

Table D-8 (Cont.): Descriptions of MODES Class Record Fields

Field	Symbolic Offset	Contents
user	MNR_MOD$L_USER_S	Count of clock ticks since system boot spent in user mode, excluding compatibility-mode time (longword,C)
compatibility	MNR_MOD$L_COMPAT_S	Count of clock ticks since system boot spent in compatibility mode (longword,C)
idle	MNR_MOD$L_IDLE_S	Count of clock ticks since system boot spent executing the NULL process (longword,C)

D.5.5 PAGE Class Record

The PAGE class record contains data describing the operation of the page management subsystem. The PAGE class record has a record type of 3 and a size of 65 bytes.

Figure D-9 illustrates the format of the PAGE class record. Table D-9 describes the fields in the data block for the PAGE class record. See Section D.5.1.1 for more information on the class header.

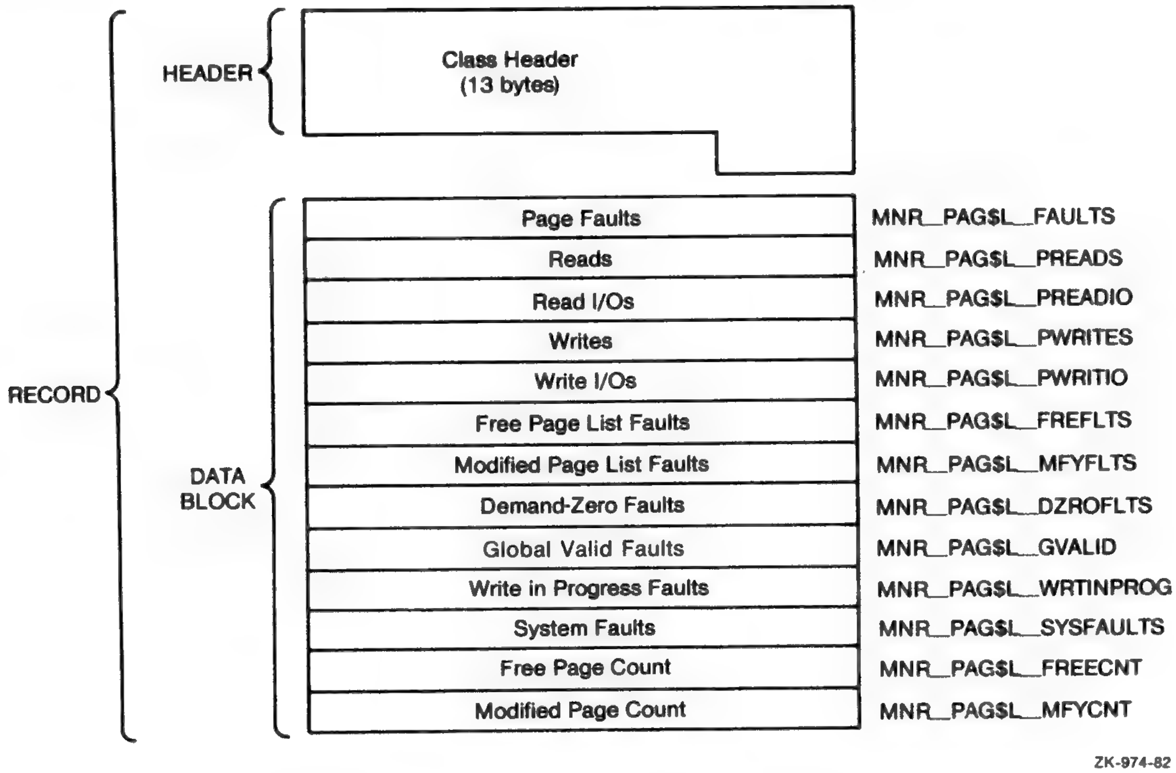


Figure D-9: PAGE Class Record Format

Table D-9: Descriptions of PAGE Class Record Fields

Field	Symbolic Offset	Contents
page faults	MNR_PAG$L_FAULTS	Count of page faults for all working sets (longword,C)
reads	MNR_PAG$L_PREADS	Count of pages read from disk as a result of page faults (longword,C)
read I/Os	MNR_PAG$L_PREADIO	Count of read I/O operations from disk as a result of page faults (longword,C)
writes	MNR_PAG$L_PWRITES	Count of pages written to the paging file (longword,C)
write I/Os	MNR_PAG$L_PWRITIO	Count of write I/O operations to the paging file (longword,C)
free page list faults	MNR_PAG$L_FREFLTS	Count of pages read from the free page list as a result of page faults (longword,C)
modified page list faults	MNR_PAG$L_MFYFLTS	Count of pages read from the modified page list as a result of page faults (longword,C)
demand-zero faults	MNR_PAG$L_DZROFLTS	Count of zero-filled pages allocated as a result of page faults (longword,C)
global valid faults	MNR_PAG$L_GVALID	Count of page faults for which the references page was found to be valid in the system global page tables (longword,C)
write in progress faults	MNR_PAG$L_WRTINPROG	Count of pages read that were in the process of being written back to disk, when faulted (longword,C)
system faults	MNR_PAG$L_SYSFAULTS	Count of page faults for which the referenced page is in system space (longword,C)
free page count	MNR_PAG$L_FREECNT	Number of pages currently on free page list (longword,L)
modified page count	MNR_PAG$L_MFYCNT	Number of pages currently on modified page list (longword,L)

D.5.6 I/O Class Record

The I/O class record contains data describing the operation of the I/O subsystem. The I/O class record has a record type of 4 and a size of 69 bytes.

Figure D-10 illustrates the format of the I/O class record. Table D-10 describes the fields in the data block for the I/O class record. See Section D.5.1.1 for more information on the class header.

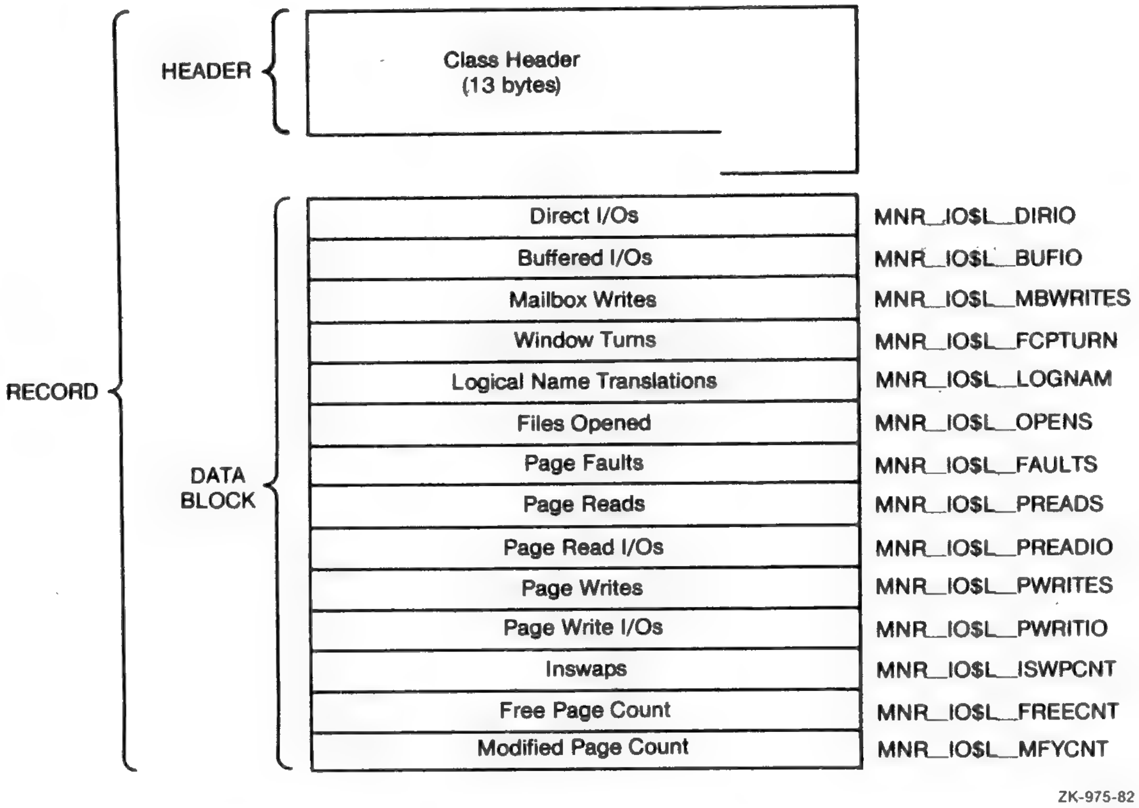


Figure D-10: I/O Class Record Format

Table D-10: Descriptions of I/O Class Record Fields

Field	Symbolic Offset	Contents
direct I/Os	MNR_IO$L_DIRIO	Count of direct I/O operations (longword,C)
buffered I/Os	MNR_IO$L_BUFIO	Count of buffered I/O operations (longword,C)
mailbox writes	MNR_IO$L_MBWRITES	Count of write-to-mailbox requests (longword,C)
window turns	MNR_IO$L_FCPTURN	Count of file mapping window misses (longword,C)
logical name translations	MNR_IO$L_LOGNAM	Count of logical name translations (longword,C)
files opened	MNR_IO$L_OPENS	Count of files opened (longword,C)
page faults	MNR_IO$L_FAULTS	Count of page faults for all working sets (longword,C)
page reads	MNR_IO$L_PREADS	Count of pages read from disk as a result of page faults (longword,C)
page read I/Os	MNR_IO$L_PREADIO	Count of read I/O operations from disk as a result of page faults (longword,C)
page writes	MNR_IO$L_PWRITES	Count of pages written to the paging file (longword,C)
page write I/Os	MNR_IO$L_PWRITIO	Count of write I/O operations to the paging file (longword,C)
inswaps	MNR_IO$L_ISWPCNT	Count of working sets read into memory from the swapping file (longword,C)
free page count	MNR_IO$L_FREECNT	Number of pages currently on free page list (longword,L)
modified page count	MNR_IO$L_MFYCNT	Number of pages currently on modified page list (longword,L)

D.5.7 FCP Class Record

The FCP class record contains data describing the operation of the file system ACPs. The FCP class record has a record type of 5 and a size of 53 bytes.

Figure D-11 illustrates the format of the FCP class record. Table D-11 describes the fields in the data block for the FCP class record. See Section D.5.1.1 for more information on the class header.

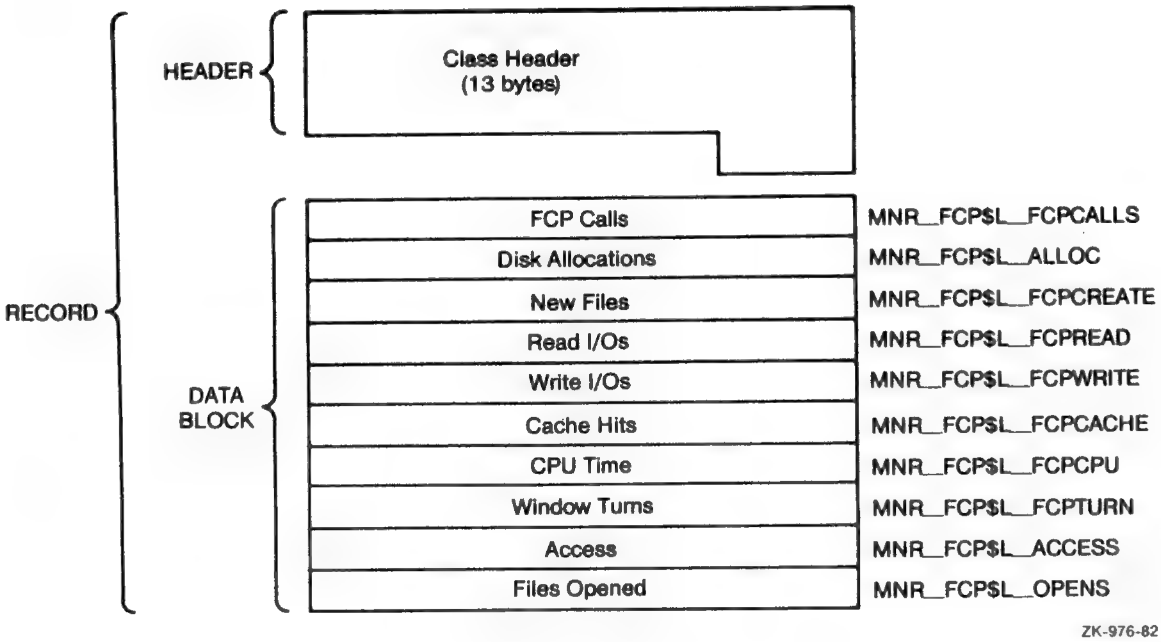


Figure D-11: FCP Class Record Format

Table D-11: Descriptions of FCP Class Record Fields

Field	Symbolic Offset	Contents
FCP calls	MNR_FCP$L_FCPCALLS	Count of QIO requests received by the file system (longword,C)
disk allocations	MNR_FCP$L_ALLOC	Count of QIO requests that caused allocation of disk space (longword,C)
new files	MNR_FCP$L_FCPCREATE	Count of new files created (longword,C)
read I/Os	MNR_FCP$L_FCPREAD	Count of read I/O operations from disk by the file system (longword,C)
write I/Os	MNR_FCP$L_FCPWRITE	Count of write I/O operations to disk by the file system (longword,C)
cache hits	MNR_FCP$L_FCPCACHE	Count of requested blocks located in the file system cache (longword,C)

(continued on next page)

Table D-11 (Cont.): Descriptions of FCP Class Record Fields

Field	Symbolic Offset	Contents
CPU time	MNR_FCP$L_FCPCPU	Count of clock ticks (10-millisecond units) of CPU time used by the file system (longword,C)
window turns	MNR_FCP$L_FCPTURN	Count of file mapping window misses (longword,C)
access	MNR_FCP$L_ACCESS	Count of file name look-up operations in file directories (longword,C)
files opened	MNR_FCP$L_OPENS	Count of files opened (longword,C)

D.5.8 POOL Class Record

The POOL class record contains data describing space allocation in the nonpaged dynamic pool. The POOL class record has a record type of 6 and a size of 45 bytes.

Figure D-12 illustrates the format of the POOL class record. Table D-12 describes the fields in the data block for the POOL class record. See Section D.5.1.1 for more information on the class header.

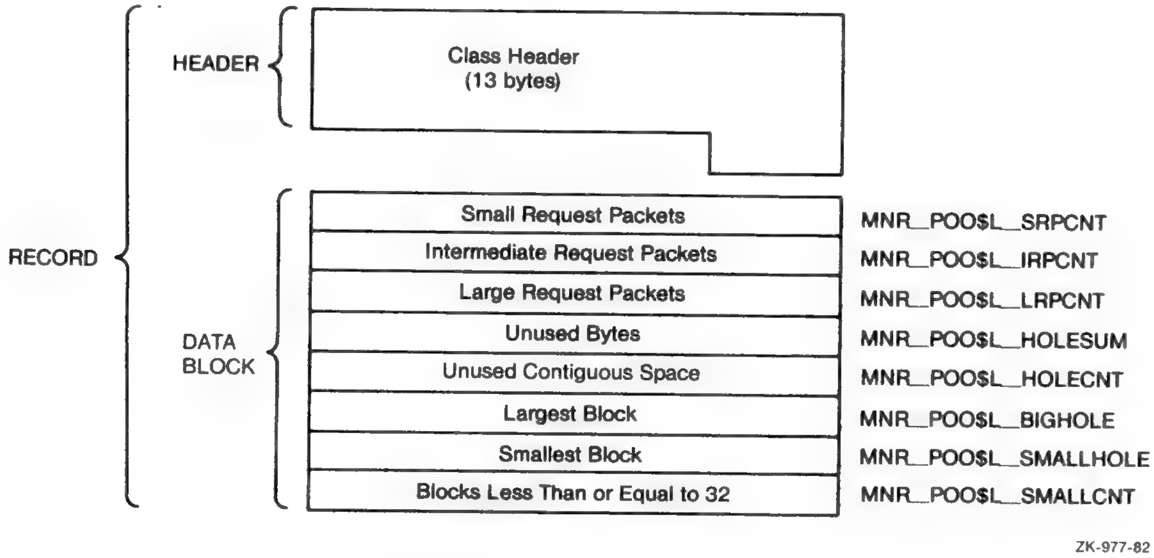


Figure D-12: POOL Class Record Format

Table D-12: Descriptions of POOL Class Record Fields

Field	Symbolic Offset	Contents
small request packets	MNR_POO$L_SRPCNT	Number of preallocated small request packets left in lookaside list (longword,L)
intermediate request packets	MNR_POO$L_IRPCNT	Number of preallocated intermediate request packets left in lookaside list (longword,L)
large request packets	MNR_POO$L_LRPCNT	Number of preallocated large request packets left in lookaside list (longword,L)
unused bytes	MNR_POO$L_HOLESUM	Total unused bytes in the dynamically allocated portion of nonpaged pool (longword,L)
unused contig space	MNR_POO$L_HOLECNT	Number of unused blocks of contiguous space in the dynamically allocated portion of nonpaged pool (longword,L)
largest block	MNR_POO$L_BIGHOLE	Size in bytes of the largest block of unused space in the dynamically allocated portion of nonpaged pool (longword,L)
smallest block	MNR_POO$L_SMALLHOLE	Size in bytes of the smallest block of unused space in the dynamically allocated portion of nonpaged pool (longword,L)
blocks less than or equal to 32	MNR_POO$L_SMALLCNT	Number of blocks less than or equal to 32 bytes in size in dynamically allocated portion of nonpaged pool (longword,L)

D.5.9 LOCK Class Record

The LOCK class record contains data describing the operation of the lock management subsystem. The LOCK class record has a record type of 7 and and size of 49 bytes.

Figure D-13 illustrates the format of the LOCK class record. Table D-13 describes the fields in the data block for the LOCK class record. See Section D.5.1.1 for more information on the class header.

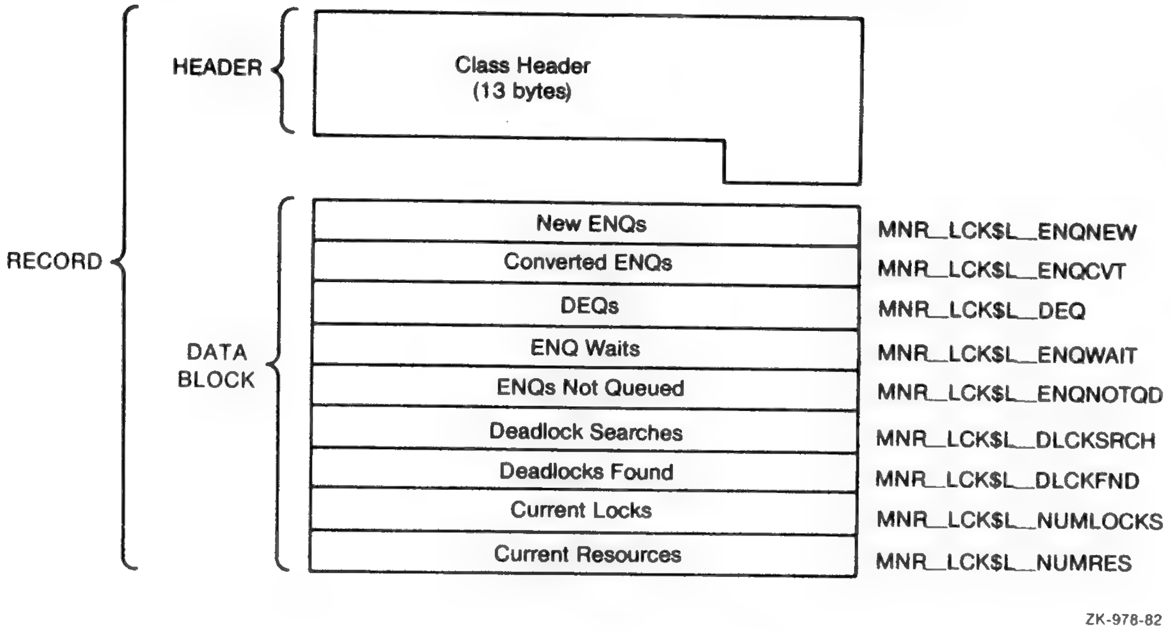


Figure D-13: LOCK Class Record Format

Table D-13: Descriptions of LOCK Record Fields

Field	Symbolic Offset	Contents
new ENQs	MNR_LCK$L_ENQNEW	Count of new ENQ (lock) requests (longword,C)
conv ENQs	MNR_LCK$L_ENQCVT	Count of converted ENQ (lock) requests (longword,C)
DEQs	MNR_LCK$L_DEQ	Count of DEQ (unlock) requests (longword,C)
ENQ waits	MNR_LCK$L_ENQWAIT	Count of times a lock could not be granted immediately, and waited (longword,C)
ENQs not queued	MNR_LCK$L_ENQNOTQD	Count of times a lock could not be granted immediately, and got error status instead of waiting (longword,C)
deadlock searches	MNR_LCK$L_DLCKSRCH	Count of times that a deadlock search was performed (longword,C)
deadlocks found	MNR_LCK$L_DLCKFND	Count of times that a deadlock was found (longword,C)
current locks	MNR_LCK$L_NUMLOCKS	Number of locks currently in the system (longword,L)
current resources	MNR_LCK$L_NUMRES	Number of resources currently in the system (longword,L)

D.5.10 DECNET Class Record

The DECNET class record contains data describing the operation of the DECnet-VAX subsystem. The DECNET class record has a record type of 8 and a size of 37 bytes.

Figure D-14 illustrates the format of the DECNET class record. Table D-14 describes the fields in the data block for the DECNET class record. See Section D.5.1.1 for more information on the class header.

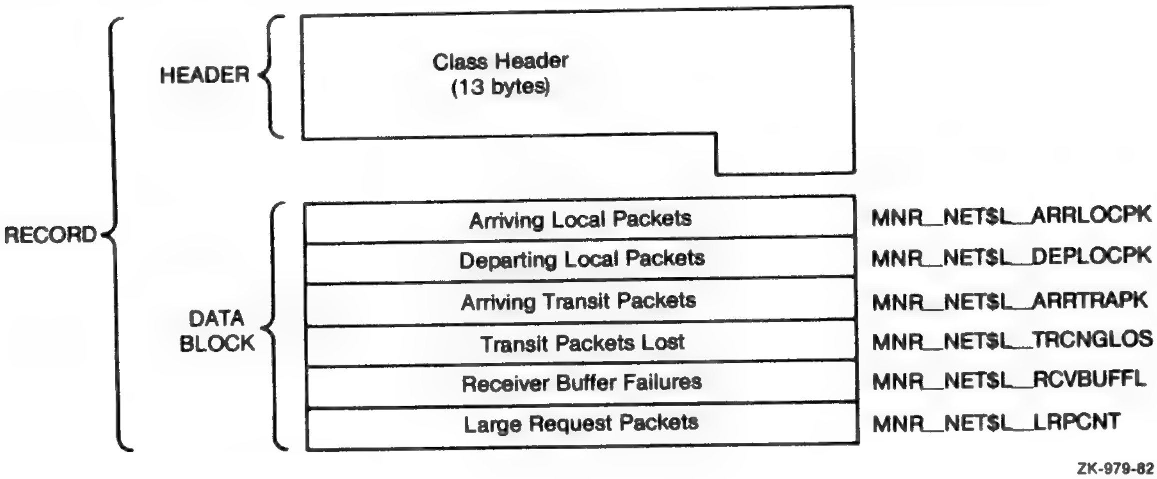


Figure D-14: DECNET Class Record Format

Table D-14: Descriptions of DECNET Class Record Fields

Field	Symbolic Offset	Contents
arriving local packets	MNR_NET$L_ARRLOCPK	Count of arriving local packets (longword,C)
departing local packets	MNR_NET$L_DEPLOCPK	Count of departing local packets (longword,C)
arriving trans packets	MNR_NET$L_ARRTRAPK	Count of arriving transit packets (longword,C)
transit packets lost	MNR_NET$L_TRCNGLOS	Count of packets lost due to transit congestion (longword,C)
rec buffer failures	MNR_NET$L_RCVBUFFL	Count of receiver buffer failures (longword,C)
large request packets	MNR_NET$L_LRPCNT	Number of preallocated large request packets left in nonpaged pool (longword,L)

INDEX

INDEX

VAX-11 Utilities
Reference Manual
AA-H781B-TE

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- □ Assembly language programmer
- □ Higher-level language programmer
- □ Occasional programmer (experienced)
- □ User with little programming experience
- □ Student programmer
- □ Other (please specify) ____________

Name ____________ Date ____________

Organization ____________

Street ____________

City ____________ State ____________ Zip Code or Country ____________

Do Not Tear - Fold Here and Tape

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

BSSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03061

Do Not Tear - Fold Here

Cut Along Dotted Line